Trust Nobody: Privacy-Preserving Proofs for Edited Photos with Your Laptop [*]

Pierpaolo Della Monica
Sapienza University of Rome, Rome, Italy
dellamonica@diag.uniroma1.it

Ivan Visconti
University of Salerno, Fisciano, Italy
visconti@unisa.it

Andrea Vitaletti
Sapienza University of Rome, Rome, Italy
vitaletti@diag.uniroma1.it

Marco Zecchini
University of Salerno, Fisciano, Italy
mzecchini@unisa.it

Abstract

The Internet has plenty of images that are transformations (e.g., resize, blur) of confidential original images. Several scenarios (e.g., selling images over the Internet, fighting disinformation, detecting deep fakes) would highly benefit from systems allowing to verify that an image is the result of a transformation applied to a confidential authentic image.

In this paper, we focus on systems for proving and verifying the correctness of transformations of authentic images guaranteeing: 1) confidentiality (i.e., the original image remains private), 2) efficient proof generation (i.e., the proof certifying the correctness of the transformation can be computed with a common laptop) even for high-resolution images, 3) authenticity (i.e., only the advertised transformations have been applied) and 4) fast detection of fraud proofs.

Our contribution consists of the following results:

- We present new definitions following in part the ones proposed by Naveh and Tromer [IEEE S&P 2016] and strengthening them to face more realistic adversaries.
- We propose techniques leveraging the way typical transformations work to then efficiently instantiate ZK-snarks circumventing the major bottlenecks due to claims about large pre-images of cryptographic hashes.
- We present a 1st construction based on an ad-hoc signature scheme and an and-hoc cryptographic hash function, obtaining for the first time all the above 4 properties.
- We present a 2nd construction that, unlike in previous results, works with the signature scheme and cryptographic hash function included in the C2PA specifications.

Experimental results confirm the viability of our approach: in our 1st construction, an authentic transformation (e.g., a resize or a crop) of a high-resolution image of 30 MP can be generated on a common 8 cores PC in about 41 minutes employing less than 4 GB of RAM. Our 2nd construction is roughly one order of magnitude slower than our 1st construction. Prior results instead either require expensive computing resources or provide unsatisfying confidentiality.

1 Introduction

There is a flourish market of web services selling high-resolution images over the Internet (e.g., Getty Images, Shutter Stock and Deposit Photos). Such markets typically work as follows. The possessor of an image I

[*]Some of the results presented in this work were initially presented in a preliminary form in September 2023 during Crossing Conference [DVVZ23] and in March 2024 at Real World Crypto Symposium [DVVZ24]. The contributions presented in [DVVZ23] also include preliminary results on a decentralized media marketplace through smart contracts, a use case only briefly mentioned in [DVVZ24].

first computes a downgraded version $\hat{I}$ of I and then makes $\hat{I}$ publicly available, keeping I private to preserve its economic value and/or to protect some sensitive content. The process to obtain a downgraded version of I consists of applying some transformations to (at least some parts of) I. For example, in some cases (e.g., adult sites and/or images of murders) images are cropped and resized in order to respect the sensitivity of possible viewers, as remarked in [KHSS22, DB23]. A user receiving and visualizing $\hat{I}$ might believe or not that $\hat{I}$ is the output of some claimed transformations on some original image, mainly depending on the reputation of the publisher of the transformed image.

The above scenario motivates the need of proving the correctness of a transformation (according to some advertised operations) of a private authentic image I into an image $\hat{I}$ that is made publicly available. The proof should be such that: a) only the advertised operations have been applied to a private authentic image I of an author identified by a public key pk and b) there is no leak of information about I.

Attested cameras and the C2PA standard. Being the author of an image associated to a public key, also a camera can be considered an author willing to guarantee the authenticity of a picture (i.e., it is not a deepfake).

Attested cameras are digital cameras with the additional capability to digitally sign a photo to ensure its authenticity. These cameras contain specific *tamper-resistant* cryptographic hardware to protect the entire signing process, (i.e., it is unfeasible to produce signed images that were not taken by the camera). Today, such cameras exist and can be purchased on the market (e.g., the Leica M11-P or the Sony Alpha 1). Moreover, with the recent interest of leading tech companies (e.g., Microsoft, Google) it is expected that this technology will be at some point available also on smartphones.

In 2021, an alliance including Adobe, Arm, Intel, Microsoft and Truepic established the project C2PA (Coalition for Content Provenance and Authenticity: `https://c2pa.org`) to create a common standard for certifying the provenance of media content. C2PA proposes a design to verify image provenance that relies on signatures produced by attested cameras. Cameras would digitally sign each photo along with a series of assertions about the photo (e.g., location, timestamp). The work of C2PA has focused primarily on how information is stored within digital media, describing its format and the algorithms to be used. Specifically, for the digital signature, the standard explicitly suggests to use SHA256 every time a cryptographic hash is involved. According to C2PA, the digital content corresponding to a picture is enriched by a data structure, called "manifest", in which the digital signature and the cryptographic hash of the image is included, plus other information (e.g., metadata).

C2PA-enabled editing applications. The C2PA specifications also includes post-processing operations through compatible editing applications. These applications append to the metadata of an edited image the original image and the editing operations. A verifier of the edited image first verifies the signature on the original image and then verifies that the operations carried out on the edited image are the ones stated and signed by the editing application. Unfortunately, as correctly observed in [DB23], using these applications raises a significant concern. It is necessary to trust the signature process within the editing software. In case an adversary extracts the signing key from the editing software (or in case there is a bug inside the software), the adversary can compute a valid signature certifying false edits, therefore circumventing the desired authenticity guarantees. Hence, C2PA-enabled editing applications are full-fledged trusted third parties, and such single points of failures can be severely exploited by adversaries.

Risks of relying on trusted third parties (TTPs). Verifying that an image $\hat{I}$ is an authentic transformation of some hidden image I created by an author is trivial in a centralized setting, relying on TTPs. In general, it is preferable to avoid or minimize the involvement of TTPs and reputation systems in order to allow a large-scale infrastructure that makes convenient and safe the access to such features (both as seller and buyer) to everyone. Indeed, beyond being potentially expensive, a TTP is a point of failure, if corrupted (e.g., under attack) it can behave maliciously, raising two obvious risks: a) the verification via a TTP could not detect a fake image; b) the confidentiality of the image I uploaded to the servers of the TTP could be compromised.

Leveraging zero-knowledge (ZK) proofs. An alternative approach through ZK proofs: the possessor of I and its signature will publish only $\hat{I}$, adding a ZK proof to certify the operations that generated $\hat{I}$ starting from an image signed by an advertised author pk. This approach is exploited in recent works of Datta and

Boneh [DB23] and of Kang et al. [KHSS22] on fighting disinformation, starting from encouraging initial results of Naveh and Tromer [NT16]. They leverage the use of succinct ZK proofs (ZK-snarks) allowing very fast verification. However, when taking into account the cryptographic hash function H used in the traditional hash-and-sign paradigm, it turns out that there are very demanding hardware requirements for generating a ZK-snark proving properties of a large pre-image of an output of H.

In order to limit those excessive requirements, the authors of [DB23, KHSS22] propose the use of ad-hoc ZK-friendly cryptographic hash functions (e.g., Lattice Hash and Poseidon Hash in [DB23], Poseidon Hash in [KHSS22]), therefore deviating from what is currently specified in the C2PA standard (i.e., SHA256). Interestingly, achieving a practical proof generation that can be computed by a cheap computer is particularly challenging, and not satisfied by [DB23, KHSS22], even using their ad-hoc cryptographic hashing that deviates from what is specified in the C2PA standard. Indeed, for the computation of such a proof, when the size of the original image I increases, their approaches require very uncommon hardware, currently available either on the cloud or through very expensive devices. This implies that the image possessor has either to make significant investments to get expensive hardware or to rely again on TTPs that have on-cloud the necessary computational resources (e.g., Amazon Web Service (AWS), Google Cloud, Azure). Both cases severely limit the large-scale secure applicability of such results. Moreover, uploading the picture to the cloud service compromises the confidentiality of the original image I.

Relaxing trust in such external services is currently an important and active research direction. For instance, Garg et al. [GGJ$^+$23], propose a solution to outsource proof computation across multiple servers guaranteeing the privacy of the prover's witness. In [CLMZ23], the authors proposed a protocol to enable a prover to outsource proof generation to a set of workers so that if some of them are honest then no private information is leaked. Both approaches leave confidentiality at risk (e.g., in case many/all servers/workers collude), and moreover, relying on the work of several respectful workers/servers leads to high costs and/or risks of unavailability. Another direction proposes to outsource the computation in a blind fashion, therefore preserving confidentiality [GGW24] but at the cost of expensive cryptographic tools.

Open Problem. The main question addressed by our work concerns the possibility of designing a system that allows the possessor of a signed image I to compute an authentic transformation $\hat{I}$ so that everyone can check the authenticity of $\hat{I}$ through a proof that informally gives the following guarantees: a) the proof preserves the confidentiality of the original image; b) the proof should be computable on a common laptop (specifically, at the time of writing, 16 GB of RAM and a CPU with 6 cores can be considered a widely spread configuration[1]); 3) the proof should guarantee authenticity of $\hat{I}$ in a strong sense, admitting a forgery only with negligible probability; 4) the proof should be very fast to verify or should at least allow an efficient-to-compute and very fast to verify fraud proof[2] (the interested reader can look at [SGB23] for a detailed discussion on fraud proofs); in other words, the (fraud) proof should be so compact and fast to verify that it could be even verified by a smart contract of a mainstream blockchain like Ethereum; 5) last, but not least, for practical relevance, we also consider C2PA requirements, and it would be beneficial if the proof can be applied to a signature computed using the cryptographic hash SHA256 that is specified in the C2PA standard.

1.1 Our Results

In this work, we provide positive answers to the above open problem by presenting a novel system for transforming images and for proving the correctness of transformations of authentic images, while preserving the confidentiality of the original image. Proofs can be computed and verified on cheap laptops, avoiding TTPs, providing also very efficient fraud proofs. Finally, differently from prior works [KHSS22, DB23], we show how convert our system in order to work with SHA256, as specified in the C2PA standard.

[1]The "Steam Hardware & Software Survey: October 2023" reports that the most popular configuration is 16 GB of RAM and 6 cores. At the time of writing, a similar device costs less than 500 EUR.

[2]In case the proof π is wrong anyone should be able to easily detect it, either because one can run an efficient verifier of π or because one can efficiently generate a fraud proof π', proving the incorrectness of π in a way that π' can be very efficiently verified.

We provide formal definitions modelling confidentiality and authenticity in natural real-world scenarios and prove the security of our system according to such definitions. The only prior work with similar contributions in terms of definitions and proofs is the one of [NT16] that, however, focuses on scenarios that differ from ours.

The core of our method evolves around the concept that given an authenticated image, one can transform it and generate a proof of the authenticity of the transformation by splitting the image into several sub-images (i.e., tiles), computing a sub-transformation and a sub-proof for every tile and, finally, using those proofs and the transformed sub-tiles to create the proof for the entire transformed image, namely the proof that $\hat{I}$ is a correct transformation of a hidden image I of an author identified through a public key pk. In this way, in contrast to previous works, we achieve a design that circumvents the need of demanding hardware due to proving computations over large pre-images of cryptographic hashes.

In summary, we show how one can efficiently compute a proof certifying the authenticity of the transformed image, preserving the confidentiality of the original image and admitting fast fraud verification. Since computing ZK-snarks over large pre-images of cryptographic hashes is required in several other scenarios (beyond image transformations), our techniques can have a strong impact in the domain of building real-world trustworthy services through cryptography, avoiding/minimizing trusted parties.

Leveraging the "tiling" technique, we propose two different constructions. The first construction, which we will call TilesProof-MT, is more efficient and allows us to demonstrate the benefits of adopting the "tiling" technique in the context of image authentication with neat improvements over prior work [DB23, KHSS22] that also focused on ad-hoc cryptographic hash functions. The second construction, which we will denote TilesProof-C2PA, works with SHA256 to be consistent with the C2PA specifications and, roughly, is one order of magnitude slower.

Proofs relying on ad-hoc cryptographic hash functions. We introduce a special signature scheme that relies on a snark-friendly cryptographic hash function, on top of which we construct the proof system TilesProof-MT. By exploiting the tiling technique, TilesProof-MT allows computing proofs much more efficiently (in terms of time and memory) with respect to the state-of-the-art.

Proofs over SHA256. The TilesProof-C2PA construction focuses on SHA256 (as in C2PA) when computing the cryptographic hash of the image, and works on high-resolution images (e.g., 30 MP) even on cheap hardware (e.g., 8 cores and 4 GB of RAM). Instead, prior works [KHSS22, DB23] discarded SHA256 for being snark-unfriendly and thus strongly deviated from the C2PA specifications. The idea behind TilesProof-C2PA is to exploit the iterative nature of SHA256. Indeed, SHA256 works in rounds and, in each round, the algorithm hashes a portion of the input message, which in our scenario is the authenticated image. We devise to compute a proof for each round (or set of rounds) using as input only the necessary portion of the image.

Performance evaluation. We have benchmarked our systems focusing on three very common image transformations: a resize, which maintains the proportions of the image and reduces the original resolution, a grayscale and a rectangular crop. These transformations are also considered by [DB23]. We used a very popular ZK-snark framework equal to the one used by [DB23], using Groth16 [Gro16] as the underlying ZK-snark. We have evaluated the performance of our implementation on a large image of 30 megapixels (6000×4000 pixels) as proposed by [DB23]. It turns out that our approach drastically reduces the memory and time consumption to generate the proof, making this task also achievable on a "common" device[1], only employing about 4 GB of RAM. This is in contrast with prior work [DB23, KHSS22] relying on cloud infrastructures to perform the same operation. The verification of our proof is fast, and the proof is compact when considering concrete scenarios. Theoretically speaking, while asymptotically the size of our proof cannot be claimed to be succinct, our proof admits a succinct and very efficient to generate/verify fraud proof. The resulting performance nicely fits natural real-world scenarios. Moreover, thanks to the work of [GMN22] it is also possible to reduce the verification time and the size of our proof. TilesProof-MT is about one order of magnitude faster than TilesProof-C2PA.

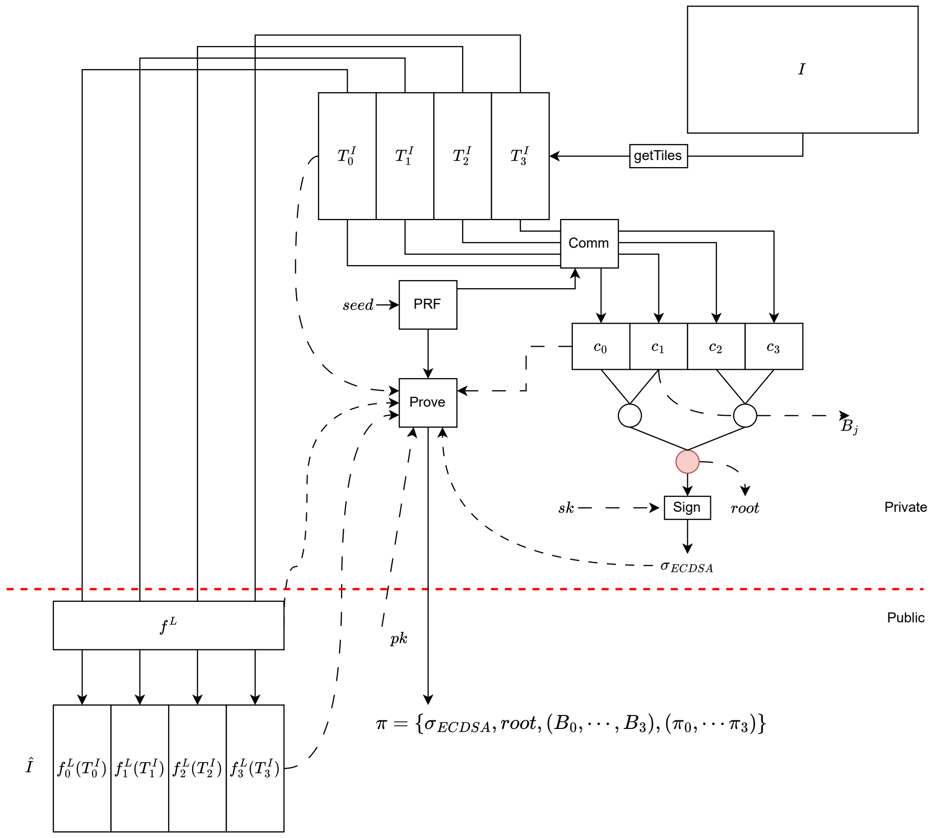


Figure 1: Graphic representation summarizing the components of TilesProof-MT. Given an image I split in 4 tiles and a transformation f^L, it shows which are the main functions, the public and private inputs for building a proof. Note that Prove embeds the ZK-snark provers to compute $(\pi_0, \cdots, \pi_3)$.

1.2 Technical Overview

We give now a high-level description of our work. The interested reader can refer to Fig. 1 for a picture summarizing the components of the TilesProof-MT construction, and Fig. 2 for the TilesProof-C2PA construction.

The tiling approach. Given an image I, we divide it into non-overlapping tiles $T_1^I, \cdots, T_n^I$ that, when displayed next to each other, correspond exactly to I. It is known that computing a ZK-snarks proving knowledge of pre-images of cryptographic hash functions becomes very unpractical when the size of the pre-image is too large. Therefore, the size of the tile will be small enough to allow a sufficiently fast computation of a ZK-snark on a cheap and widely spread computer (in our experiments we used a 16 GB of RAM and 8 cores computer, but only 4 GB of the available memory have been actually used for our tasks), but large enough to reduce the number of tiles (and in turn the number of ZK-snarks to compute and verify). Since a tile could be relatively small, its content could be predictable and thus a cryptographic hash of it might not be hiding. As such, to guarantee confidentiality, we will use a hash-based[3] commitment for each tile.

An ad-hoc signature scheme. Following prior works [DB23, KHSS22] that proposed ad-hoc hash func-

[3]For efficiency we use this commitment scheme that is secure in the random oracle model. The ZK-snark will prove a claim over an hash function used to instantiate the random oracle. In the end, we require that the instantiated hash-based commitment scheme is secure as it is (i.e., without random oracles). There are already various results relying on similar assumptions (e.g., [FW24] about Schnorr signatures). We stress that we are *not* assuming that there is a random oracle using at the same time the circuit of the hash function that instantiates the random oracle in a claim of a ZK-snark. We discuss extensively about it in Sec. 4.1.2.

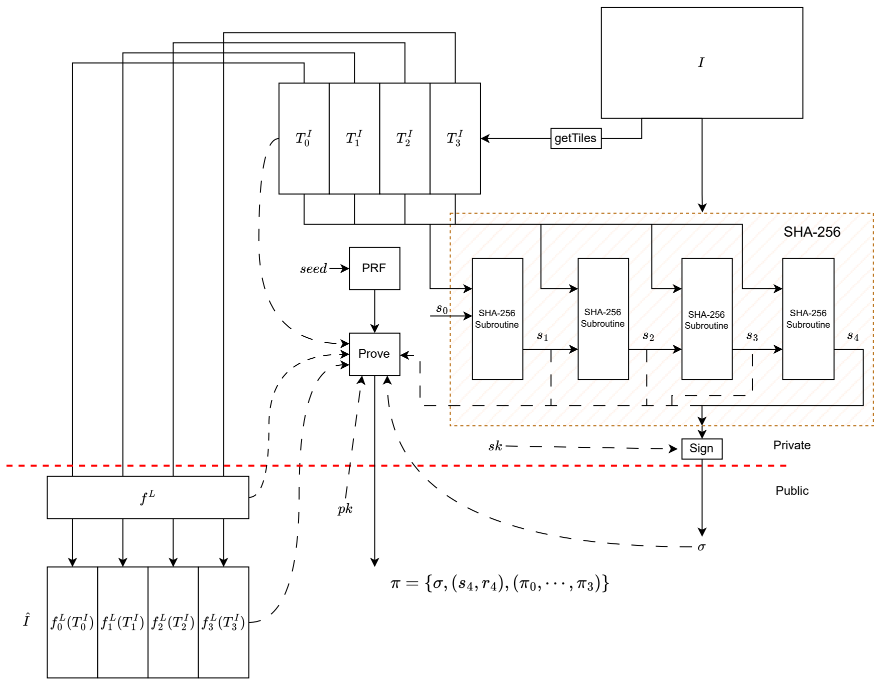


Figure 2: Graphic representation summarizing the components of TilesProof-C2PA. Given an image I split in 4 tiles and a transformation f^L, it shows which are the main functions, the public and private inputs for building a proof. Note that Prove embeds the ZK-snark provers to compute $(\pi_0, \cdots, \pi_3)$.

tions, we also present an ad-hoc hash function inside an ad-hoc signature scheme that is a variation of ECDSA. In our ad-hoc signature scheme we will include the commitments of the tiles as leaves of a Merkle Tree (MT) and its root will be signed, producing σ', using a regular signature scheme, such as ECDSA, which for concreteness we will adopt from now on. The signature σ, according to our scheme, therefore consists of σ' and the randomnesses $(r_1, \ldots, r_n)$ used to compute commitments. In order to keep σ compact, one can generate the randomness needed by the i-th commitment as $r_i = PRF(seed, i)$, and thus a compact representation of the signature $\sigma = (\sigma', r_1, \ldots, r_n)$ will simply be a pair $(\sigma', seed)$. Whenever confidentiality of the signed message is required (as in our system for authentic image transformations), the randomnesses $(r_1, \ldots, r_n)$ (and, of course, the $seed$) must remain private. The corresponding verification procedure of a signature in our scheme is kind of obvious. The message is parsed as an image and divided into tiles. The randomnesses (possibly derived from $seed$) are used to recompute all commitments, and thus the same root of the MT. Finally, the ECDSA signature of the root is verified. Jumping ahead, verifying the authenticity of a transformed image will not include the verification of the signature as specified above, indeed such a procedure would access the randomnesses used in the commitments, compromising confidentiality. However, the possession of a correctly verifiable signature will be crucial for proving through ZK-snarks the authenticity of a transformed image preserving privacy.

Proofs through ad-hoc cryptographic hashes. We consider a prover that on input some common parameters (i.e., the setup to compute a ZK-snark) and the witness (i.e., the image I and the signature σ consisting of the ECDSA signature σ' and randomnesses of commitments) will construct a proof that consists of the signature σ' of the root of the MT, and all Merkle proofs allowing to open all leaves. In other words, commitments c_j of tiles are revealed but not opened. Moreover there will be a ZK-snark π_j for every commitment c_j. The statement proved by π_j is that the transformation on the (hidden) tile T_j^I committed in c_j produces a tile $\hat{T}_j^I$ of $\hat{I}$. The crucial observation here is that the prover of the ZK-snark is invoked on input a small witness (the size is limited by the size of a tile of I), therefore, the proof involving a pre-image of the cryptographic hash used to compute c_j is not very demanding and can be computed without expensive hardware. As such, dividing the image into tiles allows us to deflate the complexity of the prover of a ZK-snark, using the tile size as the parameter to tune the computational effort according to the locally available resources. The proof π will therefore include also all π_j computed above.

The verification of the proof consists first of verifying the signature of the root of the MT and the Merkle proofs that reveal the leaves. Next, it is required that the ZK-snarks associated with each π_j in π are accepting. If any of the above checks fails, the verification process outputs 0, and it outputs 1 otherwise. The benefit of the MT is that in some cases (e.g., crop) one can construct a more efficient proof omitting some branches. This would allow skipping ZK-snarks π_j for every tile T_j^I that has no impact on $\hat{I}$.

The privacy-preserving proof over SHA256. The C2PA standard specifies ECDSA with SHA256 as signature scheme. Recall that SHA256 (following the Merkle-Dåmgard construction) works in rounds and splits the message (in our case, the image) in chunks of 512 bits, outputting for each round a string of 256 bits. Assuming that the message is split into m chunks, in the first round the algorithm combines an initial hash value of 256 bits s_0 with the first chunk of the message, outputting s_1, using a one-way compression function. We denote such a compression function by $\mathsf{SHA\text{-}256Compression}$. Then, in every intermediate round $i > 1$, it combines the output s_{i-1} of the $(i-1)$-th round with the i-th chunk, outputting s_i, always using $\mathsf{SHA\text{-}256Compression}$. The output of the last round s_m is the final hash computed by the SHA256 algorithm. Note that, in the last round, the chunk might contain some padding that has been added to the original message.

The output of SHA256 is used inside the ECDSA signature generation, thus obtaining the signature σ.

We propose the construction $\mathsf{TilesProof\text{-}C2PA}$ leveraging the "tiling" technique to realize a proof system sticking (differently from [KHSS22, DB23]) with SHA256. This is done by: **1)** dividing I into tiles $T_1^I, ..., T_n^I$ according to the computational capabilities of the prover. Each tile is constructed by grouping one or more chunks used in the rounds of SHA256; **2)** constructing a proof that consists of one ZK-snark π_j for every tile T_j^I, one commitment z_j for every T_j^I, representing the intermediate output of SHA256, and an additional ZK-snark π' along with a commitment z'. π' proves that the signature σ (committed in z' and kept hidden) on I is correct according to pk. Indeed, the signature might reveal I since σ is computed on the hash of the image. More precisely, $\mathsf{TilesProof\text{-}C2PA}$ works as follows:

- There are $\pi_1, \ldots, \pi_n$ proofs for one relation and an additional proof π' for a second relation (both relations differ from the ones of $\mathsf{TilesProof\text{-}MT}$).

  - Each π_j proves that **a)** the transformation on a (hidden) tile T_j^I outputs a tile $\hat{T}_j^I$ of $\hat{I}$ and **b)** given the (hidden) intermediate state s_{j-1} of SHA256, committed in z_{j-1}, and given T_j^I, computing several times[4] the SHA256 one-way compression function produces the (hidden) state s_j, committed in z_j.
  - π' proves that given the (hidden) signature σ, committed in z', and given the (hidden) final state s_n, committed in z_n, of SHA256 over I, the signature σ is correctly verified using the public key pk.

  Again, also in this construction, the prover of each ZK-snark is invoked on input a small witness, and thus the ZK-snarks can be computed efficiently.

- The verification of the proof consists first of running the verifier of the ZK-snark associated with each $\pi_1, \ldots, \pi_n$ and π'. If any of the above checks fails, the whole verification outputs 0, and it outputs 1 otherwise.

The transformations. Our approach relies on projecting the transformation of an entire picture I into transformations applied to tiles of I. This is the key ingredient to achieve an efficient proof generation. In

[4]With several times here we mean that, given that each tile is of dimension $512 \times k$, the prover claims that starting with the state s_{j-1} and performing k times the one-way compression function of SHA256 to process T_j^I, then the new state is s_j.

our case, there will be multiple sub-proofs (i.e., ZK-snarks) computed on the local transformations of the tiles rather than a unique proof (as done in previous work) on the whole signed image and the corresponding transformed one. While this is extremely positive for efficiency, the use of local transformations can introduce some usability issues. In general, there are no guarantees that a transformation applied to the whole image can be obtained as a combination of the same transformation applied to the tiles. This is evident considering the crop. Applying the same crop function to the whole image (e.g., a crop in the central region) and to the tiles will produce a completely different result (e.g., tiny crops of the central region of each tile). However, in practice, the above issues are easy to tackle and further details can be found in App. 5.3, where we analyze the implications of our approach showing that in most scenarios the above technicalities can be successfully addressed and that our approach is largely beneficial for several transformations that are used in the considered application scenarios.

The fraud proof. While proofs generated by our system are already pretty compact and fast to verify (see Sec. 5 for details), from a theoretical perspective, a very large number of tiles might impact negatively on proof size and verification time. To circumvent this problem, we can use results of [GMN22] to aggregate the proofs reducing the proof size and the verification time. Moreover, a non-accepting proof π is possible only in the following specific cases: a) wrong signature of the root of the MT, b) the verification of a ZK-snark fails, or c) a Merkle proof is wrong. As such, one can directly get a succinct and fast to compute/verify fraud proof that just points to the the specific part of π that fails.

Note that in both constructions the size of the fraud proof is constant, but in TilesProof-MT the verification time is logarithmic in the number of tiles while it is constant in TilesProof-C2PA.

Summary of the enjoyed properties. Our constructions satisfies confidentiality for the following reasons:

1. In TilesProof-MT, the revealed signature σ' of the root of the MT and the Merkle proofs do not reveal anything about I since the leaves of the MT consist of commitments $c_1, \cdots, c_n$ of the tiles $T_1^I, \cdots, T_n^I$ of the image I. Therefore, by the hiding property of the commitment scheme (and the pseudorandomness of the PRF), and the ZK of the ZK-snarks $\pi_1, \cdots, \pi_n$ we have that I remains indistinguishable from any other image that is compatible with $\hat{I}$.

2. In TilesProof-C2PA, the hiding of commitments $z_1, \ldots, z_n$, z' and the ZK of the ZK-snarks $\pi_1, \cdots, \pi_n, \pi'$ guarantee that I remains indistinguishable from any other image that is compatible with $\hat{I}$.

In both schemes, the knowledge soundness of the underlying ZK-snarks, the binding of the commitments, the unforgeability of the signature scheme and the collision-resistance of the hash functions guarantee that $\hat{I}$ is correctly computed, according to a known transformation, from a hidden image I signed with respect to pk (e.g., the camera or the author). This guarantees that our approach achieves authenticity. Since in TilesProof-MT we can propose our own signature scheme, we can prove the authenticity of a transformation also against adaptive adversaries, while similarly to [NT16] in TilesProof-C2PA we stick with non-adaptive adversaries.

Our approach, in contrast to prior work, is appealing allowing one to compute proofs with a common laptop, and we now summarize the key points. First of all, computing a regular signature σ is fast. Second, in our case, the computation of a ZK-snark is sufficiently efficient by design, since we split the full image into small enough tiles precisely because the ZK-snark to compute on each of them must remain efficient according to the available resources. Our design provides efficient verification of proofs and very fast verification of fraud proofs, since, as mentioned before, verifying a fraud proof consists of verifying no more than a single ZK-snark or an ECDSA signature or a Merkle proof.

Finally, we observe that the work performed to compute a proof can be in part recycled when computing other proofs on the same original image. Indeed, some of the involved ZK-snarks could be re-used when, some tiles are transformed according to the same operation. We see this speed up as a feature of our systems since it is controlled by the possessor of the signed original image that can decide to share multiple and linkable transformations of the same signed image.

We define the authenticity of the output of a transformation through a proof-of-knowledge property and the confidentiality of the image used as input to the transformation through an image-indistinguishability property. Such guarantees already suffice in several natural applications. Additional properties (e.g., non-malleability) that have been investigated generically for proof systems are not in the scope of this paper (since they are not problematic for our use cases) but can be worthy to explore in future research aiming at applying our techniques to other scenarios.

1.3 Related Work

Photoproof [NT16]. Naveh and Tromer in [NT16] introduced the concept of image authentication based on cryptographic proofs. Similarly to our results, they focus was on defining a methodology that, given an image authenticated through a digital signature, allows one to apply a set of efficiently computable transformations that along with proofs can assess the authenticity of the transformed image. They in particular address issues such as hiding the specific transformation and provide mechanisms to generate updated proofs. Their definition and construction suffer from significant limitations compared to ours. We will discuss such differences more in depth in Section 3.

Image authentication leveraging multimedia security. As also specified in [NT16], the problem of image authentication supporting permissible transformations is widely studied in the scientific literature. Previous solutions to [NT16] are not based on cryptography and can be categorized in two families of approaches: watermarking and robust hashing (e.g., [ZWZY13]). However, as already discussed in [NT16], such approaches either work for a limited set of permissible transformations or are vulnerable to an adversary who is familiar with the authentication method. In general, these techniques have non-negligible error probabilities (i.e., they too easily allow false alarms or false acceptances) due to the statistical nature of the verification algorithm.

In our work, we consider only techniques that guarantee a negligible error probability The interested reader can refer to Sec. I.B of [NT16], specifically to Table I.

The works of [DB23] and [KHSS22]. Recently, Datta and Boneh in [DB23] and Kang et al. in [KHSS22], have proposed methodologies to address the same problem outlined in [NT16]. In particular they focus on carefully instantiating the cryptographic hash function so that their ZK-snarks, can be computed having as witness a larger original image compared to [NT16]. Unfortunately, in case of high-resolution images, the computations of the prover are too expensive, and thus they must be outsourced to a third-party cloud infrastructure severely negatively affecting confidentiality and decentralization, thus limiting their applicability at-scale.

The work of [LHC$^+$23]. Li et al. in [LHC$^+$23] notice that there are notable cases where transformations only impact a small region of an image. Therefore, their work, that is concurrent to ours, shows that running just one single ZK-snark on that single subimage affected by the transformation achieves a performance improvement compared to the involvement of the full image. However, as considered by [DB23, KHSS22], there are several natural transformations that must receive as input the entire images (e.g., grayscale). This would require in [LHC$^+$23] to use the entire original image as witness of the ZK-snark computation, therefore cancelling their performance improvement. As already discussed for [DB23, KHSS22], such a proof generation requires heavy hardware assumptions. Moreover, beyond the limited scenario where their proposal gives some efficiency improvement, they only perform a performance comparison with the old results of [NT16], without discussing recent results of [DB23, KHSS22].

The work of [BFGV$^+$23]. Balbas et al. in [BFGV$^+$23] notice that the performance of general-purpose proof systems deteriorates on large inputs and ad-hoc solutions lack modularity. For these reasons, they propose a framework combining good performance with the versatility of general-purpose proof systems. In their paper, they focus on image transformations, specifically on convolution, which is an operation widely adopted in machine learning. Given that convolution is a complex transformation, they state that their approach performs even better on simpler transformations (e.g., resize, crop). However, for their experiment, they use a very powerful machine (8 cores Xeon-Gold-6154 at 3GHz and with 98 GB of RAM) and do not report any results on the memory consumption which is one of the most relevant metrics of our paper. Moreover, they do not consider in their experiments, which were realized to also compare with [KHSS22], the computation of a cryptographic hash function over the tested images. In general, [BFGV$^+$23] focuses on a problem that is orthogonal to our work and, for this reason, we do not include it in the performance evaluation.

Security definitions and analysis. We stress that, considering the entire related work, only the one of [NT16] provides definitions and a security analysis, therefore assessing the desired and achieved security of their system for transformations of authentic images. Our work will provide new definitions and, accordingly, a proper security analysis.

2 Known Tools and Definitions

Preliminaries. We denote the conditional probability of A given B as $\Pr[A|B]$. We denote by $\lambda \in \mathbb{N}$ the security parameter, and by $\approx$ the computational indistinguishability. If S is a finite set, we denote by $x \leftarrow S$ the process of sampling x from S, and by $x \overset{\$}{\leftarrow} S$ a random and uniform one. A function negl is negligible if it vanishes faster than the inverse of any polynomial (i.e., for any constant c for sufficiently large λ it holds that $\mathsf{negl}(\lambda) \leq \frac{1}{\lambda^c}$).

Collision-Resistant Hash Function. Given the practical flavor of our work, we will consider unkeyed cryptographic hash functions with a fixed output length ℓ. Therefore, the hash function will be $H : \{0,1\}^* \to \{0,1\}^\ell$. In the wild, there are several unkeyed cryptographic hash functions (e.g., SHA-256, Poseidon) that are believed to be collision-resistant since colliding pairs are unknown and seemingly hard to find. Such functions are typically used as heuristic instantiations of random oracles.

Commitment schemes. In our paper, we use a random-oracle-based commitment scheme where the message m is concatenated to a randomness r and given in input to a random oracle that outputs the commitment.

Definition 1 *A commitment scheme for the message $m \in \mathcal{M}$ is a tuple of 2 PPT algorithms $\Upsilon = (\mathsf{Commit}, \mathsf{Open})$ that works as follows and satisfies the notions of* Correctness, Hiding *and* Binding.

1) *$c \leftarrow \mathsf{Commit}(m)$ is a randomized algorithm that takes as input the message $m \in \mathcal{M}$ and, using a randomness r yields the output commitment c for the message.*

2) *$\{0,1\} \leftarrow \mathsf{Open}(c, m, r)$ is a deterministic algorithm that takes as input the message m, a string r and a commitment c, and outputs 1 when accepting the commitment, otherwise outputs 0, rejecting it.*

Correctness*: For and any message $m \in \mathcal{M}$:*

$$Pr\left[\ \mathsf{Open}(c,m,r) = 1 \ \middle|\ c \leftarrow \mathsf{Commit}(m)\ \right] = 1$$

where the above r is the randomness used by Commit.

Binding*: For every PPT $\mathcal{A}$ there exists a negligible function negl such that*

$$\Pr\left[\begin{array}{c} m \neq m' \\ \wedge\mathsf{Open}(c,m,r) = 1 \\ \wedge\mathsf{Open}(c,m',r') = 1 \end{array} \middle| \ (c,m,m',r,r') \leftarrow \mathcal{A}(1^\lambda)\ \right] \leq \mathsf{negl}(\lambda)$$

Hiding*: For any $m, m' \in \mathcal{M}$*

$$\mathsf{Commit}(m) \approx \mathsf{Commit}(m').$$

For concreteness, we will use as commitment scheme the popular hash-based construction that is secure in the random oracle model, and we use Poseidon hash [GKR+21] to instantiate the random oracle. We denote the corresponding commit algorithm with $\mathsf{Commit}_\mathsf{p}$ and the open algorithm with Open_p, namely $\Upsilon = (\mathsf{Commit}_\mathsf{p}, \mathsf{Open}_\mathsf{p})$.

We will make the heuristic assumption that such instantiation of the commitment scheme is secure and will refer to the corresponding circuit in the statements proven of the ZK-snarks.

Signature Scheme.

Definition 2 *A signature scheme is a triple of PPT algorithms* $\Psi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{VerifySign})$ *that works as follows and satisfies the notions of* Correctness *and* Unforgeability*.*

1) $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *is a key generation algorithm, which, taken as input the security parameter* λ*, outputs a key pair* $(\mathsf{pk}, \mathsf{sk})$*.*

2) $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ *is a randomized algorithm, which, taken as input the private key sk and the message* $m \in \mathcal{M}$*, outputs the signature* σ *of the message* m*.*

3) $\{0,1\} \leftarrow \mathsf{VerifySign}(\mathsf{pk}, m, \sigma)$ *is a deterministic algorithm, which, taken as input the public key* pk*, a message* $m \in \mathcal{M}$ *and a signature* σ*, outputs the bit* b*. In the case where* $b = 1$ *the verification is successful, otherwise the verification is unsuccessful.*

Correctness. *Given a key pair* $(\mathsf{pk}, \mathsf{sk})$ *generated using the algorithm* $\mathsf{Gen}(1^\lambda)$*, for any* $m \in \mathcal{M}$*:*

$$Pr\left[\mathsf{VerifySign}(\mathsf{pk}, m, \sigma) = 1 | \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)\right] = 1$$

Unforgeability. *To formally define this property, we need first to introduce the experiment* $\mathsf{ExpSigForge}$*. In this experiment, an adversary* $\mathcal{A}$ *interacts with an oracle* $O_{Sign}(\mathsf{sk}, \cdot)$ *such that:*

$$\mathcal{A} \underset{\sigma}{\overset{m}{\rightleftharpoons}} O_{Sign}(\mathsf{sk}, \cdot)$$

The experiment proceeds as follows:

```
$ExpSigForge_{\mathcal{A}}^{O_{Sign}}(\lambda)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
$(m, \sigma) \leftarrow \mathcal{A}^{O_{Sign}(sk, \cdot)}(\mathsf{pk})$
If $\mathsf{VerifySign}(\mathsf{pk}, m, \sigma) = 1$ and $m$ is not among
 the requested message to the oracle $O_{Sign}(\mathsf{sk}, \cdot)$ then:
  return 1
return 0
```

Therefore, Ψ *is* unforgeable *if, for all PPT* $\mathcal{A}$*, there exists a negligible function* negl *such that:*

$$Pr[ExpSigForge_{\mathcal{A}}^{O_{Sign}}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

For concreteness, we will use *ECDSA* with algorithms denoted with $\Psi_{ECDSA} = (\mathsf{Gen}_{\mathsf{ECDSA}}, \mathsf{Sign}_{\mathsf{ECDSA}}, \mathsf{VerifySign}_{\mathsf{ECDSA}})$. It will be a building block of our signature scheme, however it can be replaced with any other signature scheme.

Merkle Tree. A Merkle Tree computed over input values $x_1, \cdots, x_n$ is a binary tree in which the input values are placed at the leaves all with the same largest depth, and the value at each internal node is the collision-resistant hash of the values of its two children (or just the hash of the left child if the right child is missing). If n is not a power of 2, the right part of the tree will have missing nodes. The height of the tree is logarithmic in the number of leaves. The root of the MT is a succinct representation of the entire sequence $x_1, \ldots, x_n$.

Definition 3 *Given an unkeyed collision-resistant hash function defined as* $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$*, a MT construction consists of 3 PPT algorithms* $M = (\mathsf{Build}_{\mathsf{MerkleTree}}, \mathsf{ExctractLeaf}, \mathsf{VerifyLeaf})$*, such that:*

1) $(MT, root) \leftarrow \mathsf{Build}_{\mathsf{MerkleTree}}(x_1, \cdots, x_n)$ *is a deterministic algorithm, which takes a set of leaves* $x_1, \cdots, x_n$*, outputs the root and MT using* H *as collision-resistant hash function.*

2) $B_i \leftarrow \mathsf{ExctractLeaf}(MT, x_i)$ *is a deterministic algorithm, which takes as input a MT MT and a leaf x_i, and outputs a list B_i called* Merkle path *that contains, the leaf itself and all the sibling traversing the tree from the leaf to the root.*

3) $\{0,1\} \leftarrow \mathsf{VerifyLeaf}(x_i, rt, B_i)$ *is a deterministic algorithm, which takes a leaf x_i in the MT, the root rt and a Merkle path B_i and outputs a bit b. In the case where $b = 1$, verification is successful, namely x_i is a leaf of a MT with root rt. Otherwise, ($b = 0$) the verification is unsuccessful.*

We adopt a MT as a collision-resistant hash function of a message m divided into n chunks $x_1, \cdots, x_n$. Therefore, writing $\mathsf{Build}_{\mathsf{MerkleTree}}(m)$ is equivalent to writing $Build_{MerkleTree}(x_1, \cdots, x_n)$ if m is divided into n chunks.

Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-snark). We first introduce the definition of polynomial relationship and then the definition of a ZK-snark.

Definition 4 *A polynomial relation is a function $R : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}$ such that (i) $R(x,w) = 1$ implies that $|w| \leq \mathsf{poly}(|\mathsf{x}|)$ and (ii) (x,w) allows one to efficiently verify whether $R(x,w) = 1$.*

Definition 5 *A ZK-snark for an auxiliary input distribution $\mathcal{Z}$ and for a polynomial relation R is a triple of PPT algorithms $\Sigma = (\mathsf{KeyGen}, \mathsf{Prove}, \mathsf{VerifyProof})$ that satisfies the notions of* Completeness, Knowledge Soundness, Zero Knowledge *and* Succinctness *and works as follows:*

1) $\mathsf{crs} \leftarrow \mathsf{KeyGen}(1^\lambda)$ *outputs a common reference string (CRS)* crs *composed by an evaluation key and a verification key.*

2) $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w)$ *on input a CRS* crs*, an instance x and a witness w such that $R(x,w) = 1$, outputs a proof π.*

3) $\{0,1\} \leftarrow \mathsf{VerifyProof}(\mathsf{crs}, x, \pi)$ *on input a CRS* crs*, an instance x and a proof π, outputs a bit b. The verification is considered successful when $b = 1$.*

Completeness*: For any pair (x,w) such that $R(x,w) = 1$*

$$\Pr\left[\ \mathsf{VerifyProof}(\mathsf{crs}, x, \pi) = 1 \ \middle|\ \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w) \end{array} \right] = 1$$

Knowledge Soundness*: Σ has knowledge soundness for the auxiliary input distribution $\mathcal{Z}$, if for every PPT $\mathcal{A}$ there exists a PPT* extractor *algorithm* Ext *such that the following probability is at most* $\mathsf{negl}(\lambda)$

$$\Pr\left[\ \mathsf{VerifyProof}(\mathsf{crs}, x, \pi) = 1 \wedge R(x,w) = 0 \ \middle|\ \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{KeyGen}(1^\lambda) \\ \mathsf{aux}_Z \leftarrow \mathcal{Z}(\mathsf{crs}) \\ (x,\pi) \leftarrow \mathcal{A}(\mathsf{crs}, \mathsf{aux}_Z)\ ; w \leftarrow \mathsf{Ext}(\mathsf{crs}, \mathsf{aux}_Z) \end{array} \right]$$

Zero Knowledge (ZK)*: Σ satisfies (composable) zero-knowledge if there exists a simulator $\mathcal{S} = (\mathcal{S}_{kg}, \mathcal{S}_{prv})$ such that the following conditions hold for all PPT adversaries $\mathcal{A}$:*

Keys Indistinguishability:

$$\Pr\left[\mathcal{A}(\mathsf{crs}) = 1 \middle|\ \mathsf{crs} \leftarrow \mathsf{KeyGen}(1^\lambda)\ \right] \approx \Pr\left[\mathcal{A}(\mathsf{crs}) = 1 \middle|\ (\mathsf{crs}, td_k) \leftarrow \mathcal{S}_{kg}(1^\lambda)\ \right]$$

Proof Indistinguishability: for all (x,w) s.t. $R(x,w) = 1$

$$\Pr\left[\ \begin{array}{c} \pi \leftarrow \mathsf{Prove}(\mathsf{crs}, x, w), \\ \mathcal{A}(\mathsf{crs}, \pi) = 1 \end{array} \ \middle|\ (\mathsf{crs}, td_k) \leftarrow \mathcal{S}_{\mathsf{kg}}(1^\lambda)\ \right] \approx \Pr\left[\ \begin{array}{c} \pi \leftarrow \mathcal{S}_{\mathsf{prv}}(\mathsf{crs}, td_k, x), \\ \mathcal{A}(\mathsf{crs}, \pi) = 1 \end{array} \ \middle|\ (\mathsf{crs}, td_k) \leftarrow \mathcal{S}_{\mathsf{kg}}(1^\lambda)\ \right]$$

Succinctness*: The verifier runs in time* $\mathsf{poly}(\lambda + |x| + \log(|w|))$ *and the proof size is* $\mathsf{poly}(\lambda + \log(|w|))$.

Extractability in Knowledge Soundness

In the notion of knowledge soundness defined in Def. 5, following [CFQ19] we considered an auxiliary input aux_Z that is generated from a distribution $\mathcal{Z}$ that may depend on crs. Note that, knowledge soundness is impossible for some distributions of $\mathcal{Z}$, as shown in [BP15]. Still there are benign auxiliary input distributions for which the impossibility does not hold. As shown recently in [GKO$^+$23] in the random oracle model knowledge soundness, along with succinctness, is possible for every auxiliary-input distribution.

Hence, we need to precisely formalize which auxiliary inputs cannot ensure knowledge extractability: if $\mathcal{A}$ receives from aux_Z an accepting proof π for an instance x on the relation R, then the Ext cannot extract the witness w from the adversary.

2.1 Definitions from Naveh et al. [S&P 2016]

We now recall notions from [NT16], that introduced image authentication based on cryptographic proofs.

Image. Let $\mathcal{I}_{N,M}$ be the set of all possible images of dimension $N \times M$. An image $I \in \mathcal{I}_{N,M}$ is a pixel matrix $I \in \{0, 1, ..., 255\}^{3\times N\times M}$ of size $N \times M$ where the RGB values of each pixel are specified.

Original image. Let $\Psi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{VerifySign})$ be a signature scheme. Given an image I, a public key pk and a signature σ, we say that I is *original* with respect to pk if $\mathsf{VerifySign}(\mathsf{pk}, I, \sigma) = 1$.

Transformation. Given two sets of images $\mathcal{I}_{N,M}$ and $\mathcal{I}_{\hat{N},\hat{M}}$, an *image transformation* is a deterministic function $f : \mathcal{I}_{N,M} \to \mathcal{I}_{\hat{N},\hat{M}}$ (e.g., resize, crop, grayscale). We denote by Π a polynomial-size set of transformations.

3 Our New Definition

Authentic image. Given an image transformation f and a transformed image $\hat{I}$, we say that $\hat{I}$ is *authentic* with respect to a public key pk and f, if there exist I and σ such that $\hat{I} = f(I)$ and $\mathsf{VerifySign}(\mathsf{pk}, I, \sigma) = 1$.

Tile. The pixel matrix of an $N \times M$ image I can be split into n sub-matrixes, $T_1^I, \cdots, T_n^I$ each one representing a *tile* of I. We denote with $\mathsf{getTiles}(I, n)$ the function dividing the pixel matrix of I into n sub-matrixes (i.e., tiles) where each sub-matrix is of $T_{size} = \lceil \frac{N\times M}{n} \rceil$ pixels. For the sake of simplicity, we assume wlog that the number of rows and columns of a sub-matrix T_j^I is $\lceil \sqrt{T_{size}} \rceil$.

Global and local transformations. A global transformation $f^G(I)$ is a transformation applied to the whole image I. A local transformation $f^L(I)$ combines a set of local sub-transformations $f_1^L(T_1^I), \cdots, f_n^L(T_n^I)$ applied on the tiles of an image I. In other words, $f^L(I) \leftarrow \|_{j=1}^{n} f_j^L(T_j^I)$ where $\|$ denotes the operation[5] that combines the transformed tiles to obtain a transformed image.

Image-hiding proof system. Here we present our definition of a proof system that can guarantee the authenticity of a transformed image protecting the confidentiality of the original image. Our new definition only in part follows the one of [NT16]. There are some critical differences that we will discuss later in this section.

In an $\mathsf{Image\text{-}Hiding}$ proof systems, an instance for the prover $\mathsf{IHProve}$ and verifier $\mathsf{IHVerify}$ is a triple $x = (\hat{I}, \mathsf{pk}, f^L)$ where pk is a public key of a signature scheme Ψ, f^L is a local transformation and $\hat{I}$ is the output of f^L on input an image I. The corresponding witness of x for $\mathsf{IHProve}$ is $w = (I, \sigma)$ where I is the original image that is then transformed into $\hat{I}$ and σ is a signature of I generated with Ψ using the secret associated to pk.

The relation R on top of which the CRS generator $\mathsf{IHSetup}$, the prover $\mathsf{IHProve}$ and the verifier $\mathsf{IHVerify}$ are built is defined as $R((\hat{I}, \mathsf{pk}, f^L), (I, \sigma)) = 1$ if and only if $(\mathsf{VerifySign}(\mathsf{pk}, I, \sigma) = 1 \wedge f^L(I) = \hat{I})$. The relation R_{FP} for fraud proofs is $R_{FP}((\mathsf{crs}, \hat{I}, \mathsf{pk}, f^L, \pi), \pi_{\mathsf{FP}}) = 1$ if and only if $\mathsf{IHVerify}(\mathsf{crs}, (\hat{I}, \mathsf{pk}, f^L), \pi) = 0$ where crs is the CRS. The CRS, generated by $\mathsf{IHSetup}$, is composed of a set of polynomial-size sub-CRSs one for each possible $f^L \in \Pi$. Wlog, we implicitly assume that an algorithm of an $\mathsf{Image\text{-}Hiding}$ proof system

[5]In some cases this operation consists of a simple concatenation, in other cases more adjustments might be needed.

receiving the composed CRS as input selects the sub-CRS related to the transformation f^L that must be processed.

Definition 6 *Given a signature scheme* $\Psi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{VerifySign})$*, the tuple of PPT algorithms* $\Phi = (\mathsf{IHSetup}, \mathsf{IHProve}, \mathsf{IHVerify}, \mathsf{IHFPSetup}, \mathsf{IHFPProve}, \mathsf{IHFPVerify})$ *is an* Image-Hiding *proof system over* Ψ *for an auxiliary input distribution* $\mathcal{Z}$ *and for a set of transformations* Π *if for all corresponding relations* R *it satisfies* Completeness, Proof of Knowledge, Image Indistinguishability *and* Fraud Proof Succinctness *and works as follows:*

1) $\mathsf{crs} \leftarrow \mathsf{IHSetup}(1^\lambda)$ *outputs a CRS* crs.

2) $\pi \leftarrow \mathsf{IHProve}(\mathsf{crs}, x, w)$ *on input a CRS* crs*, an instance* x *and a witness* w *such that* $R(x, w) = 1$*, outputs a proof* π*.*

3) $\{0,1\} \leftarrow \mathsf{IHVerify}(\mathsf{crs}, x, \pi)$ *on input a CRS* crs*, an instance* x *and a proof* π*, outputs a bit* b*. The verification is considered successful if and only if* $b = 1$*.*

4) $\mathsf{crs_{FP}} \leftarrow \mathsf{IHFPSetup}(1^\lambda)$ *outputs a CRS* $\mathsf{crs_{FP}}$ *for* R_{FP}*.*

5) $\pi_{\mathsf{FP}} \leftarrow \mathsf{IHFPProve}(\mathsf{crs_{FP}}, x_{\mathsf{FP}}, w_{\mathsf{FP}})$ *on input a CRS* $\mathsf{crs_{FP}}$*, the instance* x_{FP} *and the witness* w_{FP} *such that* $R_{FP}(x_{\mathsf{FP}}, w_{\mathsf{FP}}) = 1$*, outputs a proof* π_{FP}*.*

6) $\{0,1\} \leftarrow \mathsf{IHFPVerify}(\mathsf{crs_{FP}}, x_{\mathsf{FP}}, \pi_{\mathsf{FP}})$ *on input a CRS* $\mathsf{crs_{FP}}$*, the instance* x_{FP} *and the proof* π_{FP}*, outputs a bit* b*. The verification is successful if and only if* b *is* 1*.*

Completeness*: For any pair* (x, w) *such that* $R(x, w) = 1$ *the following probability is equal to* 1

$$Pr\left[\mathsf{IHVerify}(\mathsf{crs}, x, \pi) = 1 \,\middle|\, \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{IHSetup}(1^\lambda) \\ \pi \leftarrow \mathsf{IHProve}(\mathsf{crs}, x, w) \end{array}\right]$$

Proof of Knowledge (PoK): Φ *has the* Proof of Knowledge *property for an auxiliary input distribution* $\mathcal{Z}$*, if for every PPT* $\mathcal{A}$ *there exists a PPT* extractor Ext *and a negligible function* negl *such that the following probability is at most* $\mathsf{negl}(\lambda)$

$$\Pr\left[\begin{array}{c} \mathsf{IHVerify}(\mathsf{crs}, x, \pi) = 1 \bigwedge \\ \hat{I} \neq f^L(I_j), 1 \leq j \leq m \bigwedge \\ (\mathsf{VerifySign}(\mathsf{pk}^*, I, \sigma) = 0 \vee f^L(I) \neq \hat{I}) \end{array} \,\middle|\, \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{IHSetup}(1^\lambda)\ ; (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ aux_Z \leftarrow \mathcal{Z}(\mathsf{crs}) \\ (x = (\hat{I}, \mathsf{pk}^*, f^L), \pi) \\ \leftarrow \mathcal{A}^{O_{Sign}(\mathsf{sk},\cdot)}(\mathsf{crs}, aux_Z, \mathsf{pk}) \\ (I, \sigma) \leftarrow \mathsf{Ext}(\mathsf{crs}, aux_Z, \mathsf{pk}, \mathsf{qt}) \end{array}\right]$$

where $\mathsf{qt} = \{I_j, \sigma_j\}$*, with* $|\mathsf{qt}| = m$*, is the transcript of all queries to the signature oracle* O_{Sign} *and its answers, specifically* I_j *is the* j*-th query and* σ_j *(i.e., the signature of* I_j *using* sk*) is the* j*-th answer.*

Fraud Proof Succinctness*:* $(\mathsf{IHFPSetup}, \mathsf{IHFPProve}, \mathsf{IHFPVerify})$ *is a snark for* R_{FP}*.*

Image Indistinguishability (ImInd)*: We first introduce the experiment ExpImageIndistinguishability. In this experiment, an adversary* $\mathcal{A}$ *interacts with a signature oracle* $O_{Sign}(\mathsf{sk}, \cdot)$ *and a transformation oracle* $O_T(\mathsf{crs}, \mathsf{sk}, I, \cdot)$ *that for a specific CRS, a specific secret key* sk *of the signature scheme and a specific image* I*, receives as input a transformation* f *and outputs* $\hat{I}$ *and a correctly computed image-hiding proof* π*:* $\mathcal{A} \underset{\hat{I},\pi}{\overset{f}{\rightleftharpoons}} O_T(\mathsf{crs}, \mathsf{sk}, I, \cdot)$*. If the adversary passes as input a function* $f \notin \Pi$*,* O_T *ignores the query.*

```
$ExpImageIndistinguishability^{O_{Sign},O_T}_{\mathcal{A},R}(\lambda)$
$\mathsf{crs} \leftarrow \mathsf{IHSetup}(1^\lambda)\ ; (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
$(I_0, I_1) \leftarrow \mathcal{A}^{O_{Sign}(\mathsf{sk},\cdot)}(\mathsf{pk}, \mathsf{crs})\ ; b \leftarrow\!\!\$\ \{0,1\}$
$b' \leftarrow \mathcal{A}^{[O_{Sign}(\mathsf{sk},\cdot);\ O_T(\mathsf{crs},\mathsf{sk},I_b,\cdot)]}(\mathsf{pk}, \mathsf{crs})$
If $f$ passed as input to $O_T$ is such that $f(I_0) \neq f(I_1)$
    $\vee\ I_0, I_1 \notin \mathcal{I}_{N,M}$ then: return 0
return $(b == b')$
```

Φ *is* image indistinguishable*, if for every PPT $\mathcal{A}$, there exists a negligible function* negl *such that the following probability is less or equal than* $\frac{1}{2} + \mathsf{negl}(\lambda)$*:*

$$Pr\left[ExpImageIndistinguishability_{\mathcal{A},R}^{O_{Sign},O_T}(\lambda) = 1\right]$$

In the ImdInd experiment, we implicitly assume that the adversary can get additional polynomial-length auxiliary input, which is chosen before the beginning of the experiment.

As for ZK-snarks, in the above definition of an Image-Hiding proof system it is still possible that a PPT prover be unpractical. In addition, it is possible that the computed proof be very long, slowing down the verifier and that computing a witness for a fraud proof and running the prover of the fraud proof could be very expensive. Later in this work we will show and analyze our constructions that (unlike prior work) addresses all the above practical issues. In our constructions the prover can be run on a common laptop, the proof is relatively short, the verifier is efficient and computing a fraud proof (including its witness) is fast.

Remarks. The PoK property considers an adversary producing an x and an accepting proof π such that x refers to a transformed image $\hat{I}$ that through π is considered authentic with respect to the public key pk^* and a transformation f^L. The experiment outputs 1 when the extractor fails, since the extractor outputs a pair (I, σ) such that either I is not consistent with the transformation f^L and with the transformed imaged $\hat{I}$ specified in x (i.e., $\hat{I} \neq f^L(I)$) or σ is not a valid signature of I according to pk^*. The extractor should fail only with negligible probability. Note that, the adversary can pick $\mathsf{pk}^* = \mathsf{pk}$ when trying to succeed with respect to a pk of a honest user, and can pick $\mathsf{pk}^* \neq \mathsf{pk}$, for a possibly maliciously generated new public key pk^*. The former case allows to model an attack in the disinformation scenario, where the adversary would like to compute an image along with an accepting proof for a fake transformation that refers to a trustworthy author (e.g., a C2PA-friendly camera[1]). The latter case allows to model an adversary in the decentralized market of digital assets, where authors of images do not necessarily have an a-priori well-established reputation and thus the adversary can choose maliciously the public key pk^*; moreover, the adversary can take advantage of existing transformed images and proofs, in order to produce an accepting proof attesting a fake transformation w.r.t. pk^*.

Both in the PoK and in the ImInd properties we could have also considered a scenario where there is a well-established CRS, and thus public keys of honest users could be adaptively computed therefore running Gen on input the CRS. However, since currently in the real world there is no standard CRS, we opted for a model where public keys of honest players are generated independently of any CRS.

We allow $\mathcal{A}$ to obtain valid signatures of images of her choice through an oracle. On top of signed images, $\mathcal{A}$ can also generate accepting proofs for transformations of such images. As such, our definition also models an adversary managing to get transformed images along with proofs.

Finally, similarly to the Knowledge Soundness property of ZK-snarks (see Sec. 2 for details), also for the PoK of a Image-Hiding proof system, in the presence of specific restrictions (e.g., when the proof is significantly shorter than the witness) there can be auxiliary input distributions such that for some adversary no extractor can succeed. This typically corresponds to an adversaries that receives $\hat{I}$ and π from the environment instead of computing them. In such cases, the extractor would need the code of whoever in the environment computed the proof in order to extract the signature and the original image (unless, as discussed for the case of knowledge soundness of ZK-snarks, there is some additional help provided by random oracles as in [GKO+23]).

Differences with definitions in [NT16]. Here we comment some important differences between the definitions proposed in [NT16] and the definition of an Image-Hiding proof system. First of all, our definition reflects real-world situations because the access to the oracles by the adversary allows to model the possibility of elaborated strategies of different types to win the game. For instance, in a real use case, the adversary can acquire at any time a C2PA-friendly camera that has previously taken the photo at sale (in the context of a decentralized marketplace) or that has been censored (in the context of disinformation) to try to obtain information on the signature of such an image (e.g., by taking other photos and studying how the signature is computed). This is mapped in our game-based definition of indistinguishability by the fact that $\mathcal{A}$ can access the signing oracle at any given moment. Furthermore, following our use-case scenarios, we see a possessor of the image (i.e., the subject that has the image I and a signature) as the party interested in producing transformations and proofs. Therefore, unlike [NT16], we do not overload our definitions with requirements about producing transformed images over already transformed images. We also strengthen the Proof-of-Knowledge property by considering a more realistic adaptive adversary accessing a signing oracle. Indeed, in [NT16], the authors devise a non-adaptive adversary producing a valid proof of a fake image only receiving auxiliary messages (i.e., a polynomial set of images and relative signatures) established before the crs generation. While we will prove that our first construction TilesProof-MT satisfies our adaptive Proof-of-Knowledge property, our second construction TilesProof-C2PA is secure only w.r.t. a non-adaptive version where pictures and their signatures are obtained upfront. Additionally, we have also considered adversarial public keys pk^* that are instead not conceived in the definition of [NT16]. In [NT16] they admit that their definition can be weak in modelling concrete scenarios and still they opted for such a weak definition since they could not prove the security of their scheme otherwise.

Our definition refines the concept of efficient verification considering applications where the succinctness of the proofs is not crucial and can be concretely compensated by efficient fraud proofs or leveraging [GMN22] to reduce verification time and proof size. As we will see later, this relaxation is beneficial to construct a practical Image-Hiding proof system where the prover is efficient and the fraud verification is very fast, nicely fitting the use-case scenarios that we have in mind. Finally, the definition of [NT16] is heavier in terms of information to maintain hidden since it considers a simulation-based definition. We relax this requirement and we protect the confidentiality of the original image with a classical game-based definition based on indistinguishability. Both adjustments turn out to be useful to improve the performance of our constructions and introduce no specific issue for the discussed use cases.

As such, we believe that our new definition is both more effective in modeling real-world scenarios while at the same time allowing to construction of very efficient schemes.

4 Constructions

Here we show Image-Hiding proof systems.

4.1 TilesProof-MT

Here we show our first and more efficient construction TilesProof-MT of a Image-Hiding proof system. In particular, we will show: a) an ad-hoc signature scheme ImageSign to sign an image; b) how to apply a local transformation to an image divided into tiles; c) the Image-Hiding proof system TilesProof-MT using ImageSign as signature scheme.

4.1.1 The ImageSign Signature Scheme

Here, we show our signature scheme ImageSign $\Psi_{IS} = (\mathsf{Gen}_{\mathsf{IS}}, \mathsf{Sign}_{\mathsf{IS}}, \mathsf{VerifySign}_{\mathsf{IS}})$. In Alg. 1, we show the $\mathsf{Sign}_{\mathsf{IS}}$ algorithm for an image I (still, the signature scheme can be used with any bit string as message space). The generation of the public and secret keys consists simply of the generation of ECDSA public and private keys. The verification of the signature can be trivially inferred by the signature algorithm, and thus, we do not report it here explicitly.

We stress that [DB23, KHSS22] proposed ad-hoc variations of the signing process relying on specific constructions of the involved cryptographic hash function (e.g., Lattice Hash + Poseidon Hash in [DB23], Poseidon Hash in [KHSS22]) deviating from what is implemented in standard cameras[1].

Note that $PRF(seed, j)$ is the output of a pseudorandom function, PRF, taking in input a *seed* and the tile index j. The PRF is used only to have a shorter representation of a full signature that otherwise would include a random string r_j for every index j. We crucially compute a commitment of (a tile of) an image to later on guarantee privacy even when the MT path is revealed. Indeed, the leaf will be a commitment and thus the underlying tile remains hidden. The use of commitments (that guarantee hiding) instead of just a cryptographic hash is another significant feature of our work in contrast to prior results [DB23, KHSS22].

Alg. 1: ImageSign Sign_{IS} algorithm for n tiles.

```
    Input: A secret key sk for ECDSA and an image $I$
1   $T^I_1, \cdots, T^I_n \leftarrow \mathsf{getTiles}(I, n)$; $seed \xleftarrow{\$} \{0,1\}^\lambda$
2   foreach tile $T^I_j$ do
3       $r_j \leftarrow PRF(seed, j)$; $c_j \leftarrow \mathsf{Commit_p}(T^I_j, r_j)$
4   $(MT, root) \leftarrow \mathsf{Build_{MerkleTree}}(c_1, ..., c_n)$
5   $\sigma_{ECDSA} \leftarrow \mathsf{Sign_{ECDSA}}(\mathsf{sk}, root)$
6   $\sigma \leftarrow (\sigma_{ECDSA}, seed)$
    /* or $(\sigma_{ECDSA}, (r_1, \ldots, r_n))$                                   */
    Output: $\sigma$
```

Computing a signature is pretty fast since it consists of an ECDSA signature, plus one evaluation of a PRF (or one sampling of a random string) and one computation of a hash-based (through Poseidon hash) commitment per tile (or more in general for each chunk of the message), plus the construction of an MT that adds no more than an evaluation of a fast cryptographic hash per tile.

Theorem 1. *Let* $\Upsilon = (\mathsf{Commit_p}, \mathsf{Open_p})$ *be a commitment scheme, let* $M = (\mathsf{Build_{MerkleTree}}, \mathsf{ExctractLeaf}, \mathsf{VerifyLeaf})$ *be a MT construction and let* $\Psi_{ECDSA} = (\mathsf{Gen_{ECDSA}}, \mathsf{Sign_{ECDSA}}, \mathsf{VerifySign_{ECDSA}})$ *be a signature scheme, then* ImageSign *is a signature scheme.*

Proof. The proof is pretty straightforward. Correctness follows by inspection. Indeed, the verification consists of building exactly the same MT starting from the image I and the randomnesses required to build the same commitments during the signature generation. As such, the roots will match and the ECDSA signature will be verified.

Unforgeability follows from the following facts. If by contradiction an adversary computes the signature of a new image (i.e., a message that was never queried to the signature oracle) with non-negligible probability p, then by the binding of the hash-based commitment, at least one leaf of the MT will be new. By the collision resistance of the MT, the signature computed by the adversary leads to computing a new root of the MT. Consequently, the signature computed by the adversary must include also a valid ECDSA signature of a new message (i.e., the new root), and this can be obviously used to build a reduction contradicting the security of ECDSA.

Notice that we have not mentioned the hiding property of the commitment scheme since it will be crucially needed (along with the pseudorandomness of the PRF) when arguing the ImInd property of our construction. □

4.1.2 ImageSign and ZK-Snarks with Oracles

In [FN16], the authors studied the feasibility of knowledge soundness in snarks when the adversary has access to oracles (this motivates the notion of O-snark). In particular, in Sec 4.3 they show that knowledge soundness still works for adversaries that have access to a signature oracle and to a random oracle when the signature scheme is in some sense O-snark friendly. Fortunately, such schemes can be obtained from traditional signature schemes following the hash-and-sign paradigm. Indeed, in [FN16] the authors propose the following tweak: 1) the hash function is modeled as a random oracle; 2) the query to the random oracle consists of the concatenation of the message to be signed and a random string r that will appear in the signature. The trick used by the extractor of [FN16] is to answer to oracle queries internally, by programming the random oracle and thus adapting some signatures hardwired in its code already at the start of the reduction. Therefore, the above tweak produces a signature scheme admitting an O-snark starting from traditional signature schemes.

One might think that ImageSign follows precisely this tweaked hash-and-sign approach because there is already a random string that is concatenated to the messages tile by tile and the hash-based commitment is anyway secure in the random oracle model. However, having an extractor programming a random oracle while the adversary aims at computing a proof over the corresponding cryptographic hash function is extremely dangerous since it corresponds to assuming at the same time that a function described by a small circuit is also uniformly random. We want to avoid this. Therefore, we do not keep ImageSign as it is, and we make sure to separate the random oracle that will be programmed by the extractor from the circuit that will be used in the claims of the ZK-snarks. Since we will be interested in ZK-snarks only about the leaves of the Merkle tree, we will apply the trick of [FN16] only to the computation of the root of the Merkle tree (i.e., the hash of the two children of the root is computed with an hash function modelled as a random oracle and a third value is added in input to this computation).

Summing up, when referring to ImageSign in the remaining part of the paper, we will implicitly assume that the root of the Merkle tree is computed by hashing through a function H_0 the concatenation of the two children and a random string that will be added to the signature. Moreover, the leaves consist of hash-based commitments, and we will assume that the commitment scheme is secure when instantiated with a cryptographic hash function H_1. As such, having a commitment (and thus the circuit of H_1) in a claim of a ZK-snark will not interfere with the fact that the extractor will program the random oracle that is instantiated with H_0.

Concluding, a signature oracle of ImageSign can be accessed by a knowledge soundness adversary and there will still be a successful extractor (related to knowledge soundness) in the random oracle model. We will use this fact when proving the PoK property of our Image-Hiding proof system TilesProof-MT, since the adversary of the PoK property of TilesProof-MT has access to an oracle answering with signatures generated according to ImageSign[6].

4.1.3 Publication of the Image Transformation

Alg. 2: Local transformation $f^L(I)$ for n tiles.

```
    Input: Image $I$, a local transformation $f^L$ (notable examples are bilinear resize, grayscale with
      luminosity method and crop).
1   $T_1^I, \cdots, T_n^I \leftarrow \mathsf{getTiles}(I, n)$
2   foreach tile $T_j^I$ do
3       $\hat{T}_j^I \leftarrow f_j^L(T_j^I)$
4   $\hat{I} \leftarrow \|_{j=0}^{n} \hat{T}_j^I$.
    Output: $\hat{I}$
```

Alg. 2 describes how to transform an image by applying local transformations to each tile. As already discussed in Sec. 1.2, instead of applying the transformation on the entire image (as in [KHSS22, DB23]), we first apply the right transformation locally to each tile and then, we reconstruct the transformed image combining the transformed tiles. In App. 5.3 we analyze in depth the implications of this approach. Alg. 2 works identically both for TilesProof-MT and for TilesProof-C2PA.

4.1.4 Proof System

We recall that the CRS generated by IHSetup is composed by a set of polynomial-size sub-CRSs one for each possible $f^L \in \Pi$. Consider the relation R_j and the generator $\mathsf{KeyGen_j}$ of the sub-CRS of the corresponding ZK-snark $\Sigma_j = (\mathsf{KeyGen_j}, \mathsf{Prove_j}, \mathsf{VerifyProof_j})$. IHSetup on input (1^λ) runs for each tile T_j^I with $j = 1, \ldots, n$ the algorithm $\mathsf{KeyGen_j}$, obtaining $\mathsf{crs_j} \leftarrow \mathsf{KeyGen_j}(1^\lambda)$ where $\mathsf{KeyGen_j}$ generates the CRS w.r.t. the j-th tile. Indeed, recall that depending on the transformation, it is possible that different tiles will end up requiring ZK-snarks related to different relations (i.e., different circuits). It goes without saying that one run of $\mathsf{KeyGen_j}$ can be recycled for all tiles that will require a ZK-snark related to the same relation. The ZK-snarks will consider the following relations $R_j((f_j^L, \hat{T}_j^I, c_j), (r_j, T_j^I))$ if and only if $(c_j = \mathsf{Commit_p}(T_j^I, r_j \wedge \hat{T}_j^I = f_j^L(T_j^I))$.

[6]We implicitly assume that ImageSign is tweaked as just described.

Generation of π. In our proof system, the prover wants to prove to the verifier that $\hat{I}$ is an authentic image

Alg. 3: The alg. Prove of TilesProof-MT for computing a proof π of I consisting of n tiles involved in the transformation.

```
Parameters: $\mathsf{crs} = \{\mathsf{crs}_1, \ldots, \mathsf{crs}_{|\Pi|}\}$, where $\mathsf{crs}_j$ is generated with $\mathsf{KeyGen_j}$ of the ZK-snark $\Sigma_j$.
Witness: Image $I$, signature $\sigma := (\sigma_{ECDSA}, seed)$ generated according to Alg. 1.
Instance: Local transformation $f^L$, transformed image $\hat{I} = f^L(I)$ computed according to Alg. 2
  and public key $\mathsf{pk}$.
1  $T^I_1, \cdots, T^I_n \leftarrow \mathsf{getTiles}(I, n)$
2  $\hat{T}^I_1, \cdots, \hat{T}^I_n \leftarrow \mathsf{getTiles}(\hat{I}, n)$
3  foreach tile $T^I_j$ do
4      $r_j \leftarrow PRF(seed, j)$; $c_j \leftarrow \mathsf{Commit_p}(T^I_j, r_j)$
5      $\pi_j \leftarrow \mathsf{Prove_j}(\mathsf{crs}_j, (f^L_j, \hat{T}^I_j, c_j), (r_j, T^I_j))$
6  $(MT, root) \leftarrow \mathsf{Build_{MerkleTree}}(c_1, ..., c_n)$
7  if $\mathsf{VerifySign_{ECDSA}}(\mathsf{pk}, root, \sigma_{ECDSA}) = 0$ then
8      Abort
9  foreach tile $T^I_j$ do
10     $B_j \leftarrow \mathsf{ExctractLeaf}(MT, c_j)$
11 $\pi \leftarrow \{\sigma_{ECDSA}, root, (B_1, \cdots, B_n), (\pi_1, \cdots, \pi_n)\}$
Output: $\pi$
```

obtained from an original signed image I where the signature can be verified using the public key pk and that the image I is divided in n tiles that have been transformed according to an advertised transformation f^L in order to obtain $\hat{I}$.

Hence, π guarantees that the prover knows every T^I_j that is used to compute the commitment c_j (that is a leaf belonging to the j-th Merkle path B_j) and to which a local transformation f^L_j is applied obtaining $\hat{T}^I_j$ that is the j-th tile of $\hat{I}$. π is composed by:

1. the ECDSA signature σ_{ECDSA} of the *root* of the MT generated according to Alg. 1;
2. the Merkle paths $B_1, \cdots, B_n$ of all nodes between the leaves (including the leaves $c_1, \cdots, c_n$) and the root;
3. the ZK-snarks $\pi_1, \cdots, \pi_n$ where each π_j proves the corresponding relation R_j.

Alg. 3 describes how π is generated. Note that through ImageSign it is possible to compute a transformation using only a portion of the original image (i.e., a subset of tiles). In this way, we can efficiently compute a crop or concatenate a crop to another transformation. Indeed, it is necessary to compute a ZK-snark π_j only for tiles involved in the transformation, as remarked in the description of Alg. 3.

Fraud Proofs. Anyone can simply build a fraud-proof for an instance $x := (\mathsf{crs}, \hat{I}, \mathsf{pk}, f^L, \pi)$, indicating in which step a proof π fails. In our construction, no additional CRS is required to compute a fraud proof (i.e., IHFPSetup outputs $\perp$). To construct a fraud-proof on input an instance that includes both the CRS used to compute the (possibly invalid) proof, the instance x (i.e., the transformed image $\hat{I} = f^L(I)$, the function f^L and the public key of the signer pk), and the (potentially invalid) proof π, the algorithm run as follows:

1) check that the signature on *root* is correct, executing $\mathsf{VerifySign_{ECDSA}}(\mathsf{pk}, root, \sigma_{ECDSA})$; if the output is 0, set $\pi_{\mathsf{FP}} = 1$ and return it; otherwise keep going;

2) check that B_j is a valid Merkle path and that c_j is a valid leaf with respect to *root* executing $\mathsf{VerifyLeaf}(c_j, root, B_j)$; if $\mathsf{VerifyLeaf}(c_j, root, B_j) = 0$, set $\pi_{\mathsf{FP}} = (2, j)$ and return it; otherwise keep going;

3) check that each π_j has been correctly computed, executing $\mathsf{VerifyProof_j}(vk_j, (f_j^L, \hat{T}_j^I, c_j), \pi_j)$; if $\mathsf{VerifyProof_j}$ $(vk_j, (f_j^L, \hat{T}_j^I, c_j), \pi_j) = 0$, set $\pi_{\mathsf{FP}} = (3, j)$ and return it; otherwise, keep going;

4) set $\pi_{\mathsf{FP}} = \bot$ and return it.

IHFPProve receives and directly outputs the witness π_{FP}, after having verified that indeed when associated to the instance $x := (\mathsf{crs}, \hat{I}, \mathsf{pk}, f^L, \pi)$, it satisfies the relation R_{FP}. Fraud proofs verification is now straightforward. If $\pi_{\mathsf{FP}} = \bot$, it means that the proof π is accepting. Otherwise, the fraud proof just refers to a single component of the original proof and there can be three cases: **1)** when $\pi_{\mathsf{FP}} = (3, j)$, the component is one of the ZK-snarks $(\pi_1, \cdots, \pi_n)$, particularly the j-th ZK-snark π_j, that is not accepted by the corresponding VerifyProof; **2)** when $\pi_{\mathsf{FP}} = (2, j)$, the component is a single MT path among $(B_1, \cdots, B_n)$, particularly the j-th MT path B_j, and the verification through VerifyLeaf fails; **3)** when $\pi_{\mathsf{FP}} = 1$, the component is the signature σ_{ECDSA} of *root* that is not verified with $\mathsf{VerifySign_{ECDSA}}$.

Completeness of the fraud proof π_{FP} follows by inspection. Knowledge soundness is obvious since the prover is just sending a pointer to public information (i.e., an element of the proof π that now is a witness) to the verifier. The running time to verify a fraud proof is bounded by the running times of a snark verifier and of a verification of a Merkle branch. Computing a witness for the fraud proof is very efficient since it just consists of running the verifier of Image-Hiding. The length of the fraud proof is constant. Summing up, (IHFPSetup, IHFPProve, IHFPVerify) is a snark, satisfying Def. 6.

Theorem 2. *Let f^L be a transformation in a set of transformations Π and let $R_1, \ldots, R_n$ be the polynomial relations associated to the n tiles of I and their local transformations f_j^L. Let $\Psi_{IS} = (\mathsf{Gen_{IS}}, \mathsf{Sign_{IS}}, \mathsf{VerifySign_{IS}})$ be the* ImageSign *signature scheme. Assuming PRF is a pseudorandom function, $\Upsilon = (\mathsf{Commit_p}, \mathsf{Open_p})$ is a commitment scheme in the standard model, $M = (\mathsf{Build_{MerkleTree}}, \mathsf{ExctractLeaf}, \mathsf{VerifyLeaf})$ is a MT construction and $\Sigma_1, \ldots, \Sigma_n$ are ZK-snarks*[7] *such that each $\Sigma_j = (\mathsf{KeyGen_j}, \mathsf{Prove_j}, \mathsf{VerifyProof_j})$ is a ZK-snark for the relation R_j, then* TilesProof-MT *is an* Image-Hiding *proof system over Ψ_{IS} for Π in the random oracle model.*

Proof. Completeness follows by inspection.

Image Indistinguishability. In this proof, we refer to a real game experiment RG as the experiment $ExpImageIndistinguishability_{\mathcal{A},R}^{O_{Sign}, O_T}(\lambda)$ played by the PPT adversary $\mathcal{A}$ accessing to the signing oracle O_{Sign} and to the oracle O_T. We remind the reader that $O_T(\mathsf{crs}, \mathsf{sk}, I, \cdot)$ is a transformation oracle that, for a specific crs, a specific secret key sk of the signature scheme and a specific image I, receiving as input a transformation f^L outputs $\hat{I}$ and a correctly computed proof π. As required by the definition, we want to show that the probability that RG ends giving in output 1 (i.e., the probability that the adversary wins) is at most $1/2 + \mathsf{negl}(\lambda)$ for infinitely many values of λ. The proof proceeds by contradiction (i.e., there exists an PPT adversary $\mathcal{A}$ winning in RG with probability at least $\frac{1}{2} + \frac{1}{\lambda^c}$, for some constant c) and uses the following hybrid experiments.

The first hybrid is $\mathcal{H}_{1\mathsf{RG}}$ and it corresponds to RG except for the generation of the crs that in $\mathcal{H}_{1\mathsf{RG}}$ is generated by a simulator $\mathcal{S}_{kg}$ and not by IHSetup. The success probability of $\mathcal{A}$ in $\mathcal{H}_{1\mathsf{RG}}$, is only negligibly far from the one in RG otherwise there is an obvious reduction that breaks the keys indistinguishability of the ZK of the ZK-snarks, therefore reaching a contradiction.

The second hybrid is $\mathcal{H}_1$ where the experiment proceeds as in $\mathcal{H}_{1\mathsf{RG}}$ except for the generation of the proof π provided by the oracle O_T along with $\hat{I}$. In particular, the ZK-snarks included in the answers of O_T in $\mathcal{H}_1$ are computed using the ZK simulator $\mathcal{S}_{prv}$. We have that the success probability of $\mathcal{A}$ in $\mathcal{H}_1$, is only negligibly far from the one in $\mathcal{H}_{1\mathsf{RG}}$ otherwise there is a direct reduction breaking the proof indistinguishability of the ZK property of a ZK-snark, therefore reaching a contradiction. More precisely, we consider $\ell + 1$ sub-hybrids where $\ell = poly(\lambda)$ is the running time of $\mathcal{A}$, and we denote them as $\mathcal{H}_1^j$ for $0 \leq j \leq \ell$ where $\mathcal{H}_1^0 = \mathcal{H}_{1\mathsf{RG}}$, $\mathcal{H}_1^\ell = \mathcal{H}_1$ and the sub-hybrids $\mathcal{H}_1^j$ and $\mathcal{H}_1^{j+1}$ (for $0 \leq j \leq \ell - 1$) differ only on the $j + 1$-th proof answered by O_T that is generated through $\mathcal{S}_{prv}$ and not through Prove. The reduction from a distinguisher between these

[7]The auxiliary input required for the security of the ZK-snarks is the same used for Image-Hiding except that it also includes a vector of message-signature pairs for distinct random messages. For simplicity we will consider the auxiliary input implicitly as essentially it "carries over".

two sub-hybrids to an adversary breaking the proof indistinguishability of the ZK definition of the $j+1$-th ZK-snark is direct. In turn, this proves the computational indistinguishability of $\mathcal{H}_{1\mathsf{RG}}$ and $\mathcal{H}_1$.

The third hybrid is $\mathcal{H}_{1-2}$. This hybrid proceeds as in $\mathcal{H}_1$ except that the output of the PRF, that it is used to compute the commitments for each tile of I_b, is replaced by pure randomness. Clearly a distinguisher among $\mathcal{H}_1$ and $\mathcal{H}_{1-2}$ violates the security of the PRF.

The forth hybrid is $\mathcal{H}_2$. In this hybrid, the experiment proceeds as in $\mathcal{H}_{1-2}$ except for the image used by O_T. In particular, given the randomly selected bit b, the image used by O_T in $\mathcal{H}_2$ is not I_b but it is I_{1-b}. The only place where there is a change in the run of the two experiments is, therefore, in the input used to compute the commitments of the leaves of the MT. Indeed, in $\mathcal{H}_{1-2}$ the commitment is computed using as input the tiles of I_b while in $\mathcal{H}_2$ it is computed using as input the tiles of I_{1-b}. Again, the success probability of $\mathcal{A}$ in $\mathcal{H}_2$, is only negligibly far from the one in $\mathcal{H}_{1-2}$. Indeed, if this is not the case, then a direct reduction breaks the computational hiding of the commitment scheme, therefore reaching a contradiction. More precisely, there are $n+1$ sub-hybrids $\mathcal{H}_2^j$ for $0 \leq j \leq n$ where $\mathcal{H}_2^0 = \mathcal{H}_{1-2}$, $\mathcal{H}_2^n = \mathcal{H}_2$ and the sub-hybrids $\mathcal{H}_2^j$ and $\mathcal{H}_2^{j+1}$ (for $0 \leq j \leq n-1$) differ only on the commitment of the $j+1$-th tile since it will be computed using as input I_{1-b} instead of I_b. The reduction from a distinguisher between these two sub-hybrids to an adversary breaking the computational hiding of the commitment in the definition is direct. In turn, this proves the computational indistinguishability of $\mathcal{H}_{1-2}$ and $\mathcal{H}_2$. Note that the experiment $\mathcal{H}_2$ still samples a bit b but then the execution continues as in $\mathcal{H}_{1-2}$ when the bit $b' = 1-b$ is sampled instead, except for the winning condition where the adversary is still required to guess b. Therefore, we can then consider similar hybrids $\mathcal{H}_{2-2\mathsf{RG}}$, $\mathcal{H}_{2\mathsf{RG}}$ and $\mathcal{H}_3$ where in $\mathcal{H}_{2-2\mathsf{RG}}$ the pure randomness used for the commitments is replaced by the output of the PRF, in $\mathcal{H}_{2\mathsf{RG}}$ the simulated ZK-snarks are replaced back by ZK-snarks computed by the ZK-snark prover, and in $\mathcal{H}_3$ the crs is again generated by the $\mathsf{IHSetup}$ algorithm. The computational indistinguishability of the outputs of these hybrids follows from the same arguments used previously, and as such we do not repeat them.

Concluding, observe that experiments $\mathcal{H}_3$ and RG are identical except that, in RG, $\mathcal{A}$ wins (i.e., the experiment ends with output 1) when $\mathcal{A}$ gives in output the bit b corresponding to the image I_b that was used for the computations of O_T during the experiment, while, in $\mathcal{H}_3$, $\mathcal{A}$ wins when giving in output the complement of the bit b corresponding to the image I_b that was used for the computations of O_T during the experiment. Since in these two experiments, the views of $\mathcal{A}$ are computationally indistinguishable, we have that a success with probability $p \geq \frac{1}{2} + \frac{1}{\lambda^c}$ (for some constant c) in RG corresponds to a success with probability $p' < \frac{1}{2}$ in $\mathcal{H}_3$. This clearly contradicts the above fact that through hybrid arguments $\mathcal{A}$ succeeds in $\mathcal{H}_3$ with non-negligible probability (i.e., it contradicts that $\mathcal{A}$, with probability $\geq 1/2 + \frac{1}{\lambda^{c'}}$ for some constant c', outputs the same bit b selected in the experiment).

Proof of Knowledge. Suppose by contradiction that for some auxiliary distribution $\mathcal{Z}$ there exists s PPT $\mathcal{A}$ of $\mathsf{TilesProof\text{-}MT}$ such that for every PPT Ext it holds that the following probability is at least $\frac{1}{\lambda^c}$ for infinitely many values of λ (i.e., $\mathcal{A}$ is successful)

$$\Pr\left[\begin{array}{c|c} & \mathsf{crs} \leftarrow \mathsf{IHSetup}(1^\lambda)\ ;(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \mathsf{IHVerify}(\mathsf{crs}, x, \pi) = 1 \bigwedge & aux_Z \leftarrow \mathcal{Z}(\mathsf{crs}) \\ \hat{I} \neq f^L(I_j), 1 \leq j \leq m \bigwedge & (x = (\hat{I}, \mathsf{pk}^*, f^L), \pi) \\ (\mathsf{VerifySign}(\mathsf{pk}^*, I, \sigma) = 0 \vee f^L(I) \neq \hat{I}) & \leftarrow \mathcal{A}^{O_{Sign}(\mathsf{sk},\cdot)}(\mathsf{crs}, aux_Z, \mathsf{pk}) \\ & (I,\sigma) \leftarrow \mathsf{Ext}(\mathsf{crs}, aux_Z, \mathsf{pk}, \mathsf{qt}) \end{array}\right]$$

where c is a non-negative constant and $\mathsf{qt} = \{I_j, \sigma_j\}$, with $|\mathsf{qt}| = m$, is the transcript of all queries to the signature oracle O_{Sign} and its answers, specifically I_j is the j-th query and σ_j (i.e., the signature of I_j using sk) is the j-th answer.

In the following, we will show that for a specific extractor Ext, the above non-negligible probability allows us to reach a contradiction. Let n be the number of tiles associated to f^L. We have that π includes n ZK-snarks $\pi_1, \ldots, \pi_n$. Ext will embed the extractor[8] Ext' that exists from the knowledge soundness of the

[8]For simplicity we are wlog considering the case where all n ZK-snarks are computed for the same relation and thus the same extractor works for all of them. In general, Ext includes extractors for all relations associated to the transformation specified in π.

ZK-snark. As discussed in subsection 4.1.2, $\mathcal{A}$ has access to an oracle and this can jeopardize extractability. However, we designed Ψ_{IS} to be O-snark friendly for the specific oracle available to $\mathcal{A}$. In particular, this allows Ext to internally simulate the signature oracle O_{Sign} by programming the random oracle used to compute the root of the MT that belongs to a signature of Ψ_{IS}. Therefore, as proved in [FN16], extraction from the ZK-snarks in π by running internally Ext′, fails only with negligible probability and Ext obtains a valid witness w_j for each of the claims proved by the ZK-snarks $\pi_1, \ldots, \pi_n$. Indeed, if this is not the case, then there is a reduction breaking the assumption that the ZK-snark enjoys knowledge soundness for auxiliary distribution $\mathcal{Z}$ also in the presence of O_{Sign} (which in turn, as discussed in subsection 4.1.2 would contradict the knowledge soundness of the ZK-snark for the specific auxiliary input distribution $\mathcal{Z}$, without oracles).

Recall that a witness w_j is a pair (T_j^I, r_j) that produces the commitment c_j and moreover is such that $f_j^L(T_j^I) = \hat{T}_j^I$, where $c_j, \hat{T}_j^I$ and f^L are all specified in the claim proven by the j-th ZK-snark. Ext combines all T_j^I obtaining I such that $f^L(I) = \hat{I}$. Moreover, obtaining all r_j, Ext reconstructs[9] the entire randomness r for all commitments. By associating r to σ' (that is the ECDSA signature of the root of the MT and it is in π), Ext obtains $\sigma = (\sigma', r)$ and outputs (I, σ).

Given the n leaves $c_1, \ldots, c_n$ of the MT in π, we distinguish two cases depending on the following event E: a prior query $I^\star$ of $\mathcal{A}$ to the signature oracle received as answer a signature $\sigma^\star = (\sigma^{\star'}, r^\star)$ such that the leaves of the associated Merkle tree correspond exactly to $c_1, \ldots, c_n$. Since the success of the adversary is restricted to the condition that $\hat{I} \neq f^L(I^\star)$ while instead $f^L(I) = \hat{I}$ we have that there exists a c_j for $1 \leq j \leq n$ such that c_j can be opened in two different ways, one according to (I, r) and the other one according to $(I^\star, r^\star)$, violating the binding of the commitment scheme. Therefore, conditioned on $\mathcal{A}$ being successful, E can happen only with negligible probability.

We are left with the case in which conditioned on $\mathcal{A}$ being successful, the leaves of the MT do not match the ones associated to the answers of the signature oracle. By the collision resistance of the MT, except with negligible probability, the root of the MT in π will be new (i.e., different from all other roots belonging to the answers of O_{Sign}). Since $\mathcal{A}$ is successful, we have that IHVerify outputs 1, and thus in π there is a correct ECDSA signature of the new root. The fact that this is obtained with non-negligible probability contradicts the unforgeability of ECDSA and this concludes the proof of this theorem. □

Remark on transformations ignoring some tiles.

Note that through ImageSign it is possible to compute a proof of an authentic transformation using only a portion of the original image. This is possible when the transformation is applied only on a subset of the tiles (i.e. when a crop is realized). From a broader point of view, the prover can compute a proof of an authentic transformation more efficiently (i.e., by using only a subset of tiles) every time the transformation includes a crop (e.g., crop-and-resize). Note that the definition of PoK of Image-Hiding requires that the extractor obtains the whole image I and the randomnesses $r_1, \ldots, r_n$ (or the seed that generates them) so that a successful verification is possible through VerifySign. When a proof is computed on a subset of tiles, it is clear that the extractor cannot obtain I and $r_1, \ldots, r_n$ because not even the honest prover would used them to compute π, but Merkle paths bringing to the commitments of the involved tiles suffice. However, the extractor can still obtain the involved tiles and verify the authenticity of such tiles without knowing I and without using VerifySign of ImageSign but exploiting how a signature is constructed internally. To show this, for simplicity, let us suppose that a transformation is applied only on one tile (i.e., the image is first cropped, by choosing just one tile, and subsequently on the cropped image another transformation is carried out). The proof π in this case is composed of $\{\sigma_{ECDSA}, root, B_i, \pi_i\}$, namely, it is computed by considering only T_i^I. By internally running the ZK-snark extractor Ext′, Ext obtains the pair (T_i^I, r_i). Such a pair produces the commitment c_i that can not be opened to another value otherwise we contradict the binding of the commitment. Proving that c_i belongs to the Merkle Tree can be guaranteed with B_i and $root$ (that are in π) otherwise we contradict the collision resistance of the MT. Finally, a valid signature σ_{ECDSA}

[9]Recall that according to Ψ_{IS}, the randomness for the commitments of the leaves is sufficient to produce an accepting signature, the seed of the PRF is used only for compactness, not for security.

on a new *root* would contradict the unforgeability property of ECDSA.

Therefore, Ext extracts a valid witness (consisting of a single tile T_i^I along with its randomness r_i) but with respect to a slightly revisited relation. It is straightforward that the same considerations apply when more than one tile is involved.

4.2 TilesProof-C2PA

Here, we show our second construction named TilesProof-C2PA, an Image-Hiding proof system that works using the signature scheme (i.e., ECDSA) and the cryptographic hash (i.e., SHA256) of C2PA.

Signature. The C2PA specifications[10] indicate ES256 (ECDSA using P-256 as curve and SHA256 as a cryptographic hash function) as signature scheme. During the signing process, the message to be signed is hashed with SHA256, and then the ECDSA signing procedure is run on input the hashed message using the P-256 curve.

A discussion on how SHA256 works can be found in Sec. 1.2. We will refer with SHA-256Compression to the internal one-way compression function of SHA256.

Limits from ZK-Snarks with oracles. Recall that in Section 4.1.2, we already discussed the consequences of constructing an extractor for the knowledge soundness of an adversary having access to a signature oracle and a random oracle. In TilesProof-MT, we have used the tweaked hash-and-sign approach proposed by [FN16] to allow extraction along with access to a signing oracle.

When sticking with ES256 no tweak is possible and thus for the PoK of TilesProof-C2PA similarly to [NT16], we are limited to non-adaptive adversaries. More precisely, $\mathcal{A}$ receives as input polynomially many message-signature pairs (i.e., $(I_0, \sigma_0), \ldots, (I_k, \sigma_k)$ with $k = \mathsf{poly}(\lambda)$), but has no access to a signing oracle.

Construction. We recall that the CRS generated by IHSetup is composed of a set of polynomial-size sub-CRSs one for each possible $f^L \in \Pi$ plus a special sub-CRS. Let $\Sigma_j = (\mathsf{KeyGen_j}, \mathsf{Prove_j}, \mathsf{VerifyProof_j})$ be the ZK-snark for the j-th local transformation. IHSetup on input 1^λ runs $\mathsf{KeyGen_j}$ for each tile T_j^I with $j = 1, \ldots, n$, obtaining $\mathsf{crs_j} \leftarrow \mathsf{KeyGen_j}(1^\lambda)$. Obviously, one run of $\mathsf{KeyGen_j}$ can be recycled for all tiles requiring a ZK-snark for the same relation. For $j = 1, \ldots, n$, the j-th ZK-snarks works for the following relation R_j:

$$R_j((f_j^L, \hat{T}_j^I, z_j, z_{j-1})(T_j^I, s_j, s_{j-1}, r_j, r_{j-1})) \text{ if and only if}$$

$$(s_j = \mathsf{SHA\text{-}256Subroutine}(T_j^I, s_{j-1}) \wedge$$

$$z_{j-1} = \mathsf{Commit_p}(s_{j-1}, r_{j-1}) \wedge$$

$$z_j = \mathsf{Commit_p}(s_j, r_j) \wedge \hat{T}_j^I = f_j^L(T_j^I))$$

The special sub-CRS generated by IHSetup is structured for a ZK-snark $\Sigma' = (\mathsf{KeyGen}', \mathsf{Prove}', \mathsf{VerifyProof}')$, for the following relation R':

$$R'((\mathsf{pk}, z_n, z'), (r', \sigma, r_n, s_n)) \text{ if and only if}$$

$$(z_n = \mathsf{Commit_p}(s_n, r_n) \wedge z' = \mathsf{Commit_p}(\sigma, r') \wedge$$

$$\mathsf{VerifySign_{ES256}}(\mathsf{pk}, s_n, \sigma) = 1)$$

Generation of π. According to the above relations, a proof π guarantees that the prover knows that: a) T_j^I is the j-th input, along with s_{j-1}, to the j-th SHA-256Compression that outputs s_j; b) the commitments of s_{j-1} and s_j are, respectively, z_{j-1} and z_j; c) a local transformation f_j^L is applied to T_j^j obtaining the j-th tile of $\hat{I}$ (i.e., $\hat{T}_j^j$); d) pk correctly verifies the signature σ that is computed over s_n (i.e., the SHA256 of I) and committed in z'.

[10]https://c2pa.org/specifications/specifications/1.0/specs/C2PA_Specification.html#_digital_signatures

Detailed description of the prover is given in Alg. 4. More in details, in Alg. 4 at line 6 we used SHA-256Subroutine to represent the execution of multiple rounds of SHA-256Compression. We recall that SHA-256Compression is a one-way compression function, namely it represents a single round execution of SHA256 (i.e., given 512 bits of message and 256 bits of state, it outputs 256 bits corresponding to the new state). Given as input a tile T_j^I and a state s_{j-1}, SHA-256Subroutine executes the SHA-256Compression function $\lceil |T_j^I|/512 \rceil$ times and output a new state s_j (where the size of the input tile is defined according to the prover's computational capabilities in terms of hardware).

A proof π consists of:

1. the ZK-snarks $\pi_1, \cdots, \pi_n$ where each π_j proves the corresponding relation R_j.
2. the ZK-snark π' proving the relation R';
3. the commitments $(z_1, \cdots, z_n)$ (i.e., the commitments of $(s_1 \cdots, s_n)$) and z' (i.e., the commitment of the signature σ).

The construction of the fraud proof for TilesProof-C2PA is even simpler than the one of TilesProof-MT (see Sec. 4.1.4) since it consists of identifying and giving in output a single invalid ZK-snark proof.

The security of TilesProof-C2PA follows that of the TilesProof-MT, here we give only a brief discussion of the main deviations compared to the proof of Th. 2.

ImgInd is proved following mutatis mutandis the proof of TilesProof-MT, without non-trivial deviations.

As previously remarked, differently from TilesProof-MT, the PoK property of TilesProof-C2PA that we prove considers a non-adaptive adversary (i.e., signatures of images can not be asked during the experiment but are obtained by the adversary upfront). The proof of non-adaptive PoK starts as the proof of PoK for TilesProof-MT, assuming by contradiction that an adversary succeeds and showing an extractor that runs the underlying extractors of the ZK-snarks obtaining all committed values along with their randomnesses. Since by contradiction the adversary succeeds, by the knowledge soundness of the ZK-snarks one of the following cases must happen with non-negligible probability: 1) for some $j \in \{1, \ldots, n\}$ the message s_j committed in z_j and extracted from the $(j+1)$-th ZK-snark is different from the one extracted from the j-th ZK-snark; this breaks the binding of the commitment scheme; 2) case 1 did not happen and the extracted message-signature pair is accepting and the message is not among the signed images received upfront; this breaks the unforgeability of ECDSA with SHA256.

5 Experimental Results

Here we describe our experiment showing that the most demanding component of our system, namely the computation of a ZK-snark for a sufficiently large tile size, can be carried out in reasonable time on our "common" hardware[1]: an Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz processor with 8 cores and 16 GB of RAM, employing only about 4 GB of the available memory for our tasks.

We have developed six Circom circuits, available on GitHub[11], to compute the proofs at line 5 of Alg. 3 and lines 8 and 9 of Alg. 4, with the purpose of evaluating their performance. Our experiments consider as input to the proof three very common transformations: bilinear resize[12], rectangular crop and grayscale based on the luminosity method[13]. Our experiments confirm, according to [KHSS22], that irrespectively from the transformation, the computation of the cryptographic hash is by far the most expensive circuit component. In view of the results on the resize/crop/grayscale function, we expect a similar performance with other simple transformations, in terms of time and memory consumption.

[11] https://github.com/PIERdemo/Privacy-PreservingProofs4EditedPhotos
[12] https://chao-ji.github.io/jekyll/BilinearResize.html
[13] https://mmuratarat.github.io/rgb_to_grayscale

Alg. 4: The alg. Prove of TilesProof-C2PA for computing a proof π for I consisting of n tiles.

```
Parameters: $\mathsf{crs} = \{\mathsf{crs}_1, \ldots, \mathsf{crs}_{|\Pi|}, \mathsf{crs}'\}$, where $\mathsf{crs}_j$ is generated with $\mathsf{KeyGen_j}$ of the ZK-snark $\Sigma_j$,
  for the $j$-th tile, and $\mathsf{crs}'$ is generated with $\mathsf{KeyGen}'$ of the ZK-snark $\Sigma'$.
Witness: Image $I$, signature $\sigma$ generated according to the ES256 signing algorithm.
Instance: Local transformation $f^L$, transformed image $\hat{I} = f^L(I)$ computed according to Alg. 2
  and public key $\mathsf{pk}$.
1  $T_1^I, \cdots, T_n^I \leftarrow \mathsf{getTiles}(I, n)$
2  $\hat{T}_1^I, \cdots, \hat{T}_n^I \leftarrow \mathsf{getTiles}(\hat{I}, n)$
3  $seed \xleftarrow{\$} \{0,1\}^\lambda$
4  $r' \leftarrow PRF(seed, 0)$ ; $z' \leftarrow \mathsf{Commit_p}(\sigma, r')$
5  foreach tile $T_j^I$ do
       /* $s_0$ is the initial hash value */
6      $s_j \leftarrow \mathsf{SHA\text{-}256Subroutine}(s_{j-1}, T_j^I)$
7      $r_j \leftarrow PRF(seed, j)$ ; $z_j \leftarrow \mathsf{Commit_p}(s_j, r_j)$
8      $\pi_j \leftarrow \mathsf{Prove_j}(\mathsf{crs}_j, (f_j^L, \hat{T}_j^I, z_j, z_{j-1}),$
9                      $(T_j^I, s_j, s_{j-1}, r_j, r_{j-1}))$
10 $\pi' \leftarrow \mathsf{Prove}'(\mathsf{crs}', (\mathsf{pk}, z_n, z'), (r', \sigma, r_n, s_n))$
11 $\pi \leftarrow \{(\pi_1, \cdots, \pi_n), (z_1, \cdots, z_n), z', \pi'\}$
Output: $\pi$
```

5.1 Main Technical Choices

We used Circom to write the circuit, Snarkjs to set up the proof and Rapidsnark to generate the proof.

We instantiated TilesProof-MT with Poseidon 128 to implement the commitments of the tiles. For the evaluation of the circuit of the ZK-snark we encoded the input according to the approach proposed by Khovratovich[14]. In our experiments, we used the Circom implementation of Poseidon 128, and we set the number of field elements of a single Poseidon Sponge iteration to 16, which is the maximum value available. For all the other parameters, we refer the reader to the Circom official repository[15].

We instantiated the circuit of the ZK-snark in TilesProof-C2PA adopting the Circom implementation of SHA256[16]. In particular, to compute a round of SHA256 from a specific input state to a specific output state, we used the SHA256 compression function implemented in the library. To compute the commitment of the input state and of the output state, we used Poseidon 128, and thus we used its circuit for the corresponding ZK-snarks.

A natural question is whether an optimal tile size exists to speed up the computation of a proof. We observe that the time and memory consumptions during the generation of a proof that computes a Poseidon hash and a SHA256 compression are linear in the size of the input, as long as the swap memory is not activated. Indeed, when the swap is used the system performance is severely downgraded due to the inherently time-consuming nature of disk memory accesses. A tile therefore should not be so large to saturate the available RAM. In order to prove the viability of our approach on a common device, we performed experiments on a tile size such that the generation of a ZK-snark for the corresponding relation uses approximately 4 GB of memory. This low memory requirement is suitable also for a smartphone, therefore broadening the use cases and the audience of potential users.

[14] https://hackmd.io/@7dpNYqjKQGeYC7wMlPxHtQ/BkfS78Y9L
[15] https://github.com/iden3/circomlib/master/circuits/poseidon.circom
[16] https://github.com/iden3/circomlib/master/circuits/sha256/

	Tile Dimension	**Setup**		**Prove**		**Verify**	
	Pixels	Memory (GB)	Time (sec)	Memory (GB)	Time (sec)	Memory (GB)	Time (sec)
Crop	184756	14.1	5319	3.4	20.8	0.15	0.6
Resize	184756	14.1	5232	3.4	18.9	0.15	0.6
Grayscale	80000	14.7	5544	4.5	25.7	0.15	0.6

Table 1: Performance of a ZK-snark using TilesProof-MT (see Section 4.1.4).

	Tile Dimension	**Setup**		**Prove**		**Verify**	
	Pixels	Memory (GB)	Time (sec)	Memory (GB)	Time (sec)	Memory (GB)	Time (sec)
Crop	2666	14.7	3129	4.2	18.5	0.15	0.6
Resize	2666	14.7	3107	4.2	18.1	0.15	0.6
Grayscale	2666	14.7	3244	4.3	18.7	0.15	0.6

Table 2: Performance of a ZK-snark using TilesProof-C2PA (see Section 4.2).

5.2 Performance Evaluation

Here we show the results of the experiments to concretely assess the feasibility of our constructions. First we show the performance for the proof generation on crop, grayscale and resize employing only 4 GB of RAM. Then, we compare our work with the state-of-the-art solutions.

Thanks to our techniques allowing to choose a tile dimension according to the available hardware, we can compute the proof on images of the same size of [DB23, KHSS22] using a common laptop instead of using expensive resources or cloud infrastructures. Still, our experiments show that the performance of our 1st construction is affordable in computation time. The one of our 2nd construction is one order of magnitude slower, but it works on original images signed with ECDSA using SHA256 as in C2PA specifications.

Note that, the setup phase requires more than 4 GB of RAM, but still less than the 16 GB available on our "common" hardware. This operation is performed only once for a given transformation of a specific tile size, its output does not contain any private information and can be thus publicly shared. Therefore, the setup phase can also be computed on a more powerful platform or even on the cloud without violating our enforced properties.

Experiments on the TilesProof-MT. Table 1 summarizes the results of our experiments with the 1st construction.

The proving time for a single tile requires between 18.9 and 25.7 seconds for all the three transformations, and the necessary memory is below 4 GB except for the grayscale. In a grayscale transformation, input and output sizes are the same, while in our experiments both crop and resize have an output size which is half of the input one. This justifies the higher memory requirements of grayscale.

The size of a ZK-snark for a single tile is about 800 bytes and its verification requires 150 MB of RAM and about 0.6 secs which roughly is also the time to verify a fraud-proof.

Experiments on the TilesProof-C2PA. Table 2 summarizes the results of our experiments with the 2nd construction. The proving time for a single tile requires approximately the same amount of time (18 secs) for all the transformations. The size of the ZK-snark for a single tile is about 800 bytes and its verification requires 150 MB of RAM and about 0.5 secs and this does not deviate much from the cost of verifying a fraud-proof. As expected, since SHA256 is not a snark-friendly hash function, the size of the tiles is lower in this case compared to TilesProof-MT and strongly dominates the memory consumption, thus implying that in this case, to fit into the 4 GB memory constraint, all the transformations are applied to the same tile size.

Comparison with Kang et al. [KHSS22]. In Table 3, we compare the performance of our system to those presented in [KHSS22]. According to Table 4 in [KHSS22], the resize applied to an HD image of size $1280 \times 720 = 921600$ pixels requires 70.7 GB. Furthermore, the proof computation needs an AWS instance with 64 vCPU cores and 512 GB of RAM. Table 3 shows that we outperform [KHSS22] both in the time needed to generate a proof and, more importantly, on the necessary RAM, while the time to

	Prov	**Ver (FPVer)**	**Proof Size (FP Size)**	**Resources**	
ZK-IMG (Resize) [KHSS22]	328.2s	6.9 ms (N.A.)	3.04 KB (N.A)	70.7 GB Intel Xeon 8375C, 64 vCPU	☹
This paper (Resize) [TilesProof-MT]	94.5 s	3 s (0.6 s)	4 KB (800 bytes)	3.4 GB, Intel Corei7-8565U, 16 vCPU	☺
This paper (Resize) [TilesProof-C2PA]	6262 s	207.6 s (0.6 s)	276.8 KB (800 bytes)	4.2 GB, Intel Core i7-8565U, 16 vCPU	☺
ZK-IMG (Crop) [KHSS22]	328.2s	5.3 ms (N.A.)	3.04 KB (N.A)	70.7 GB, Intel Xeon 8375C, 64 vCPU	☹
This paper (Crop) [TilesProof-MT]	104 s	3 s (0.6 s)	4 KB (800 bytes)	3.4 GB, Intel Core i7-8565U, 16 vCPU	☺
This paper (Crop) [TilesProof-C2PA]	6401 s	207.6 s (0.6 s)	276.8 KB (800 bytes)	4.2 GB, Intel Core i7-8565U, 16 vCPU	☺

Table 3: Performance comparison between our work and [KHSS22] from HD to SD. We use 5 tiles of size 184756 pixels for TilesProof-MT tests. We use 346 tiles of size 2666 pixels for TilesProof-C2PA tests. Prov = time for the prover, Ver = time for the verifier, FPVer = time for the fraud proof verifier.

verify the proof is significantly higher but still absolutely acceptable in several practical scenarios. We stress that with the limited hardware requirements considered in our experiments, the system proposed in [KHSS22] would fail either for insufficient memory or for the gigantic amount of time required to compute a proof when a significant part of the required data are in the swap. Finally, notice that when the original image must be transferred to the external party providing the high-performance computing platform, image indistinguishability is clearly not satisfied.

Performances over very large images. We conducted our tests also on a high-resolution image of size 6000×4000 pixels (i.e., 30 MP) as in [DB23], even if that work does not explicitly report on the performance and some details on the experimental settings are missing. Here we focus on the performance of the resize transformation; similar results hold for the other transformations.

The image is divided into 130 tiles of size 184756. Considering the results in Table 1 the proof requires about 41 mins and at most 3.4 GB of memory. The verification time is instead 78 secs. The size of the proof including 130 ZK-snarks, 130 commitments, the root hash and the size of an ECDSA signature is around 108 KB.

The performance of TilesProof-C2PA is affected by the fact that SHA256 is not snark friendly. In our experiments we have considered 9003 tiles of size 2666 and the proof for the whole image needs about 45.2 hours and the verification time is 90 mins. The size of the proof, including 9003 ZK-snarks, 9003 commitments, the commitment of the ECDSA signature and π', is around 7.4 MB. Despite the proof requires a significant time it can be computed on 4.2 GB, thus proving the efficacy of the tiling approach also over huge images and not-friendly snark operations. Performance might be easily improved employing more RAM.

Packing together proofs with [GMN22]. In our work, it is also possible to aggregate snark proofs according to [GMN22] reducing the proof size (by a logarithmic factor in the size of the proofs to be aggregated) and the verification time (by a logarithmic factor in the number of proofs to be aggregated).

According to the benchmarking conducted in [GMN22], it is possible to verify 8192 Groth16 proofs in ~ 33 ms with a proof size of ~ 40 KB, while 16384 Groth16 proofs can be verified in ~ 58 ms with a proof size of ~ 43 KB. Since in the worst case considered in our experiments, TilesProof-C2PA needs 9003 tiles for 30 MP image, namely 9003 Groth16 proofs, applying this technique in our context will provide results within the above bounds.

5.3 The Impact of Local Transformations

Our approach relies on transformations applied to tiles (see Alg. 2) rather than to the whole image. While this is extremely positive efficiency-wise, it can introduce some usability issues. In particular, a) there are no guarantees that the quality of $f^G(I)$ is as good as the one of $f^L(I)$ and b) small changes to the parameters of f^L (e.g., in the case of rectangular crop we need to specify a different sub-area to be cut for every tile) might imply the re-computation of the involved f^L_j and the corresponding circuits and this might be a cumbersome process.

In this subsection, we argue that in many natural scenarios, locally transformed images are (essentially) equal to the images obtained by applying the global transformation. Furthermore, we discuss how local transformations impact the usability of the technique proposed in this paper.

We consider the following relevant transformations: the rectangular crop, bilinear resize and grayscale based on the luminosity method (also considered in [KHSS22, DB23], refer to those papers for more details about the transformations). A detailed description of these transformations is available in App. A. Note that, similar arguments can be extended to other transformations. Some transformations, such as rotation and/or flipping, are consistent with our definition of image indistinguishability, but since they easily allow the adversary to obtain I form $\hat{I}$, they do not have any practical sense in our scenarios[17].

In transformations working on individual pixels (e.g., grayscale) the function applied to the tiles produces a result identical to the one applied to the whole image. Moreover, the same circuit can be exploited for all the tiles.

Other transformations, instead, might require more circuits depending on the position of the tile impacting on *usability*. Moreover, applying local transformations and joining their outputs could produce a result that *differs* from the one of a global transformation.

The usability of local transformations. We remark that the usability issue of considering several circuits, dealing with the various/many parameters of each transformation, also affects previous work relying on proofs that consider the entire image. The additional and potential usability drawback of our approach is that for some transformations and parameters, one might end up needing a different circuit for each tile instead of a single circuit for the entire image.

The rectangular crop is a remarkable example of this issue (see Figure 3). Users are indeed free to crop any rectangular region; consequently, in the worst case, 10 distinct kinds of local transformations and the corresponding circuits are needed, namely: top, bottom, right, left, the 4 corners, the middle and the excluding one[18].

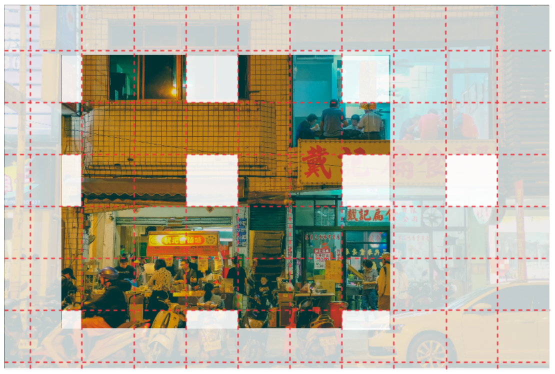


Figure 3: Number of distinct local transformations and corresponding circuits for the rectangular crop.

[17]When confidentiality of the original image is not required one can just send the original image, the signature, and the transformation to apply.

[18]The full resolution images, the transformed images and the output of the experiments are at https://github.com/PIERdemo/Privacy-PreservingProofs4EditedPhotos

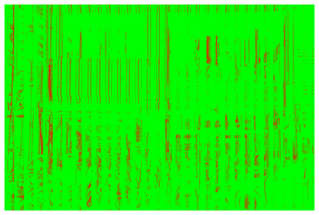
However, in some cases, the number of necessary circuits can be reduced (e.g., it can become 2 instead of 10), by selecting a crop region that matches the tiling structure.

Moreover, while tiling is beneficial in terms of proof generation and verification time, we notice that by carefully selecting the tile size, our approaches can significantly reduce the time-intensive set-up phase for the circuit. This suggests, that at least time-wise, the usability of our system remains acceptable also for the task of generating circuits, even when multiple circuits could be required.

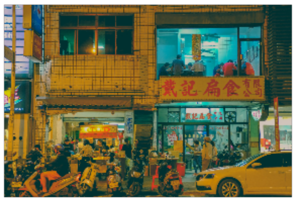
The quality of local transformations.

We performed some experiments to prove that the local bilinear resize function provides results that are hardly distinguishable by human eyes from the global ones (see Fig. 4). We have not tested the crop and the grayscale because they work pixel-by-pixel and thus there is no quality loss from local transformations. The implications from our tests on the resize operation can be applied to other transformations that work similarly to the resize (e.g., blur). We applied our local resize to a test image, and we compared the results with the ones obtained by applying a global resize. To note the difference between the two outputs, we developed[11] a filter that shows the pixels with a difference in any of the RGB channels of at least a given threshold (i.e., 5 in our test)[18]. While the filter clearly shows some differences (7% in total), we stress that they cannot be easily grasped by human eyes[18], making the final results indistinguishable in many concrete real-world scenarios.

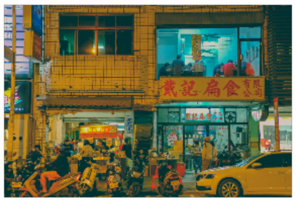
(a) Tested Image.


(b) Image highlighting the different pixels.

$f^L(I)$



$f^G(I)$



(c) Resized images with global and local transformations.

Figure 4: Result of the test for proving the indistinguishability of the resize applied to the whole image and the resize applied to the tiles.

6 Application Contexts

The properties achieved by our system allow us to extend to the mass the ability to autonomously compute authentic transformations and their proofs reducing the need of TTPs.

The fast fraud detection property is particularly useful for developing a smart contract running on renowned blockchains like Ethereum and invoked to efficiently handle the frauds and penalize malicious users, thus incentivizing correct behaviors.

However, such property can also be used to develop a system to detect fakes on the Web. Similarly to what has been proposed by [DB23], we envision a new feature allowing browsers to give an explicit sign on the authenticity of a picture, similarly to the lock for HTTPS. When an image is downloaded, it is initially unverified. The user can ask the browser to verify the proof. If it is correct, the image is classified as verified otherwise, it is classified as fake. Moreover, there can be repositories of fraud proofs that are accessed by browsers in order to check quickly, even automatically (i.e., without an explicit request of the user) if a picture is fake. This approach is to some extent similar to the verifications that are performed by the browser on Certificate Revocation Lists or for malware detection.

References

[BFGV$^+$23] D. Balbás, D. Fiore, Maria I. González V., D. Robissout, and C. Soriente. Modular sum-check proofs with applications to machine learning and image processing. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 1437–1451, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3576915.3623160.

[BP15] E. Boyle and R. Pass. Limits of extractability assumptions with distributional auxiliary input. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015*, pages 236–261, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. doi:10.1007/978-3-662-48800-3_10.

[CFQ19] M. Campanelli, D. Fiore, and A. Querol. Legosnark: Modular design and composition of succinct zero-knowledge proofs. Cryptology ePrint Archive, Paper 2019/142, 2019. doi:10.1145/3319535.3339820.

[CLMZ23] A. Chiesa, R. Lehmkuhl, P. Mishra, and Y. Zhang. Eos: Efficient private delegation of zk-SNARK provers. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 6453–6469, Anaheim, CA, August 2023. USENIX Association. URL: https://dl.acm.org/doi/10.5555/3620237.3620598.

[DB23] T. Datta and D. Boneh. Using zero-knowledge proofs to fight disinformation. In *IACR Real World Crypto Symposium (RWC)*, 2023. URL: https://iacr.org/submit/files/slides/2023/rwc/rwc2023/13/slides.pdf; https://www.youtube.com/watch?v=MwTK6ZQhOQg&t=2953s.

[DVVZ23] P. Della Monica, I. Visconti, A. Vitaletti, and M. Zecchini. Enhanced non-fungible tokens. In *CROSSING Conference*, 2023. URL: https://www.crossing.tu-darmstadt.de/media/crossing/events/crossing_conference_2023/slides_3/CROSSING_Conf_2023_Visconti.pdf; https://www.crossing.tu-darmstadt.de/news_events/conferences_workshops/crossing_conference_2023/conf_2023_schedule.en.jsp.

[DVVZ24] P. Della Monica, I. Visconti, A. Vitaletti, and M. Zecchini. Do not trust anybody: Zk proofs for image transformations tile by tile on your laptop. In *IACR Real World Crypto Symposium (RWC)*, 2024. URL: https://iacr.org/submit/files/slides/2024/rwc/rwc2024/92/slides.pdf; https://www.youtube.com/watch?v=X8ebjijCTMA.

[FN16] D. Fiore and A. Nitulescu. On the (in)security of snarks in the presence of oracles. In *Theory of Cryptography*, page 108–138, Berlin, Heidelberg, 2016. Springer-Verlag. doi:10.1007/978-3-662-53641-4_5.

[FW24] G. Fuchsbauer and M. Wolf. Concurrently secure blind schnorr signatures. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 124–160, Cham, 2024. Springer Nature Switzerland. doi:10.1007/978-3-031-58723-8_5.

[GGJ$^+$23] S. Garg, A. Goel, A. Jain, G. Policharla, and S. Sekar. zkSaaS: Zero-Knowledge SNARKs as a service. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4427–4444, Anaheim, CA, August 2023. USENIX Association. URL: https://www.usenix.org/system/files/usenixsecurity23-garg.pdf.

[GGW24] S. Garg, A. Goel, and M. Wang. How to prove statements obliviously? In *Advances in Cryptology – CRYPTO 2024*, 2024. URL: https://eprint.iacr.org/2023/1609.

[GKO$^+$23] C. Ganesh, Y. Kondi, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Witness-succinct universally-composable snarks. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 315–346, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-30617-4_11.

[GKR$^+$21] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. Poseidon: A new hash function for Zero-Knowledge proof systems. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 519–535. USENIX Association, August 2021. URL: https://www.usenix.org/system/files/sec21-grassi.pdf.

[GMN22] N. Gailly, M. Maller, and A. Nitulescu. Snarkpack: Practical snark aggregation. In Ittay Eyal and Juan Garay, editors, *Financial Cryptography and Data Security*, pages 203–229, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-18283-9_10.

[Gro16] J. Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 305–326. Springer, 2016. doi:10.1007/978-3-662-49896-5_11.

[KHSS22] D. Kang, T. Hashimoto, I. Stoica, and Y. Sun. Zk-img: Attested images via zero-knowledge proofs to fight disinformation, 2022. arXiv:2211.04775.

[LHC$^+$23] K. Li, C. Hsu, M. Chang, F. Liu, S. Chien, and W. Chen. Region-aware photo assurance system for image authentication. In *2023 IEEE 6th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 1–6, 2023. doi:10.1109/MIPR59079.2023.00037.

[NT16] A. Naveh and E. Tromer. Photoproof: Cryptographic image authentication for any set of permissible transformations. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 255–271, 2016. doi:10.1109/SP.2016.23.

[SGB23] I. Seres, N. Glaeser, and J. Bonneau. Naysayer proofs. Cryptology ePrint Archive, Paper 2023/1472. Presented in FC24, 2023. URL: https://fc24.ifca.ai/preproceedings/39.pdf.

[ZWZY13] Y. Zhao, S. Wang, X. Zhang, and H. Yao. Robust hashing for image authentication using zernike moments and local features. *IEEE TIFS*, 8(1):55–63, 2013. doi:10.1109/TIFS.2012.2223680.

A Local Transformations

The purpose of this appendix is to show how to map global transformations (resize, crop, grayscale and blur) to tiles in order to obtain a resulting locally transformed image that is (essentially) equal to the one obtained by applying the global transformation to the image (as shown in Section 5). Furthermore, we show how local transformations impact on the usability of the technique proposed in this paper.

Notations. As already discussed in Section 4, we represent an image as an RGB bi-dimensional matrix of pixels. We denote with $I[i][j]$ the pixel p at the i-th row and the j-th column of the image I. Each pixel p consists of 3 bytes. Each byte represents a color component of the pixel and we refer to it as "channel". We denote with $p.G$ the green component of the pixel, with $p.R$ the red component and with $p.B$ the blue component. To simplify the reading, when arithmetic operations or functions are applied on a pixel, we intend that these operations are separately applied to each of the 3 bytes of the pixel (e.g., with $p \times 2$ we mean $(p.R \times 2, p.G \times 2, p.B \times 2)$ or writing $max(p, p')$ is equal to $(max(p.R, p'.R), max(p.G, p'.G), max(p.B, p'.B))$).

A.1 The Algorithms of the Local Transformations

A.1.1 Bilinear Resize

In our work, we adopted the bilinear resize proposed by [DB23]. Alg. 5 shows the steps to perform such a resize on a tile T_j^I.

Alg. 5: Bilinear Resize transformation on a Tile T_j^I with x_{rows} rows and $y_{columns}$ columns.

```
Input: Tile $T_j^I$, the number of rows in output $x_{res}$, the number of columns in output $y_{res}$
1 $x_{ratio} \leftarrow (x_{rows} - 1)/(x_{res} - 1)$
2 $y_{ratio} \leftarrow (y_{columns} - 1)/(y_{res} - 1)$
3 foreach $y$ in $1, \cdots, y_{res}$ do
4     foreach $x$ in $1, \cdots, x_{res}$ do
5         $x_l, y_l \leftarrow \lfloor x_{ratio} \cdot x \rfloor, \lfloor y_{ratio} \cdot y \rfloor$
6         $x_h, y_h \leftarrow \lceil x_{ratio} \cdot x \rceil, \lceil y_{ratio} \cdot y \rceil$
7         $x_w \leftarrow ((x_{ratio} \cdot x) - x_l)$
8         $y_w \leftarrow ((y_{ratio} \cdot y) - y_l)$
          /* This next line is done for each RGB channel; we avoid explicitly writing it for
             readability */
9         $\hat{T}_j^I[x][y] \leftarrow T_j^I[y_l][x_l] \cdot (1 - x_w) \cdot (1 - y_w)$
                  $+ T_j^I[y_l][x_h] \cdot x_w \cdot (1 - y_w)$
                  $+ T_j^I[y_h][x_l] \cdot y_w \cdot (1 - x_w)$
                  $+ T_j^I[y_h][x_h] \cdot y_w \cdot x_w$
Output: $\hat{T}_j^I$, RGB matrix with $x_{res}$ rows and $y_{res}$ columns
```

A.1.2 Rectangular Crop

Alg. 6 shows the steps to perform a rectangle crop on a tile T_j^I. Among the transformations that we discuss, Crop has in general the disadvantage of requiring the computation of 10 circuits (i.e., top, bottom, left, right, 4 corners, center and excluded tile). However, by playing with the tile size it is possible to reduce the number of necessary circuits. As an example, we might set the tile size to exactly match the corner of the cropped image. In this case, we can envision a software producing transformations of an authentic picture capable of providing recommendations on the optimal tile size in view of the image in input and the crop.

Alg. 6: Rectangular Crop transformation on a Tile T_j^I with x_{rows} rows and $y_{columns}$ columns.

```
Input: Tile $T_j^I$, the starting row of the crop $x_{begin}$, the ending row of the crop $x_{end}$, the starting
 column of the crop area $y_{begin}$, the ending column of the crop area $y_{end}$ foreach $x$ in
 $x_{begin}, \cdots, x_{end}$ do
1    foreach $y$ in $y_{begin}, \cdots, y_{end}$ do
2        $\hat{T}_j^I[x - x_{begin}][y - y_{begin}] \leftarrow T_j^I[x][y]$
Output: $\hat{T}_j^I$ with $x_{end} - x_{begin}$ rows and $y_{end} - y_{begin}$ columns
```

A.1.3 Grayscale

Alg. 7 shows the steps to perform a grayscale on a tile T_j^I. The output of this algorithm is a bi-dimensional matrix G_j^I where each element is represented by a single byte.

Alg. 7: Grayscale transformation on a Tile T_j^I with x_{rows} rows and $y_{columns}$ columns

```
Input: Tile $T_j^I$
1 foreach $x$ in $1, \cdots, x_{rows}$ do
2     foreach $y$ in $1, \cdots, y_{columns}$ do
3         $p \leftarrow T_j^I[x][y]$
4         $G_j^I[x][y] \leftarrow 0.21 \cdot p.R + 0.72 \cdot p.G + 0.07 \cdot p.B$
Output: $G_j^I$ is a matrix with $x_{rows}$ rows and $y_{coloums}$ where each cell is a byte
```

A.1.4 Blur Median Transformation

Alg. 8: Blur transformation on a Tile T_j^I with x_{rows} rows and $y_{columns}$ columns.

```
Input: Tile $T_j^I$, kernel dimension $k$, the median function median
1 foreach $x$ in $1, \cdots, x_{rows}$ do
2     foreach $y$ in $1, \cdots, y_{columns}$ do
3         $\hat{T}_j^I[x][y] \leftarrow \text{median}(T_j^I[x-k][y-k], T_j^I[x-k][y-k-1], \cdots, T_j^I[x][y], \cdots, T_j^I[x+k][y+k-1], T_j^I[x+k][y+k])$
Output: $\hat{T}_j^I$ RGB matrix with $x_{rows}$ rows and $y_{coloums}$
```

Alg. 8 shows the steps to perform a blur median transformation on a tile T_j^I. In the algorithm, for each pixel p of the RGB Matrix of the tile, we build a kernel matrix taking all the pixels of a squared neighborhood of dimension $2 \cdot k \times 2 \cdot k$ with p as the center of the square. Note that if p is on the margin, there exists at least one pixel of its kernel that is not in T_j^I. For instance, $T_j^I[0][0]$ does not have all the pixels on its left and its top. In this case, we will exclude the corresponding kernel pixels from the computation of the transformation.