2023 IEEE 6th International Conference on Multimedia Information Processing and Retrieval (MIPR) | 979-8-3503-0781-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/MIPR59079.2023.00037

Region-aware Photo Assurance System for Image Authentication

Ke-Han Li[1, 4], Chih-Fan Hsu[1], Ming-Ching Chang[2], Feng-Hao Liu[3],
Shao-Yi Chien[4], and Wei-Chao Chen[1]

[1]Inventec Corporation, [2]University at Albany, [3]Washington State University, [4]National Taiwan University

Abstract

Zero-knowledge Proof (ZKP) allows active image authentication to prove image integrity after editing without revealing its source. However, existing ZKP solutions require impractical execution time, leaving a considerable gap between theory and practice. To tackle this, we present a Region-Aware Photo Assurance System based on the nature of image editing with privacy: sensitive information is usually local and relatively small, and thus by cropping and/or adding mosaic to these small regions suffices to protect privacy. Using ZKP for the locally edited region and digital signature for the others can still ensure the integrity of an image, with significantly better efficiency. We comprehensively analyzed the system's performance and showed the advantage of our system compared with the state-of-the-art ZKP-based method, PhotoProof, with 15x/60x faster on the KeyGen/Proof operations and 25x lower in the Proof size. Furthermore, we protect several real-world images selected in the Redaction dataset with our system. Our system achieves up to 2,700x faster than PhotoProof for a proof generation. We expect the system can become a practical system for real-world applications.

1 Introduction

The development of multimedia and social media has increased the sharing of images and videos [1]. Despite convenience, digital contents can easily be modified in malicious ways, incurring new challenges of misinformation and disinformation [21]. Image authentication methods [3] are developed to verify the origin, integrity, and authenticity of images. Generally, image authentication methods can be organized into *passive* and *active* methods. Passive methods [5] detect whether an image has been modified based on pixel information. However, the passive methods are sophisticated and can not guarantee the authentication results for unseen manipulation methods.

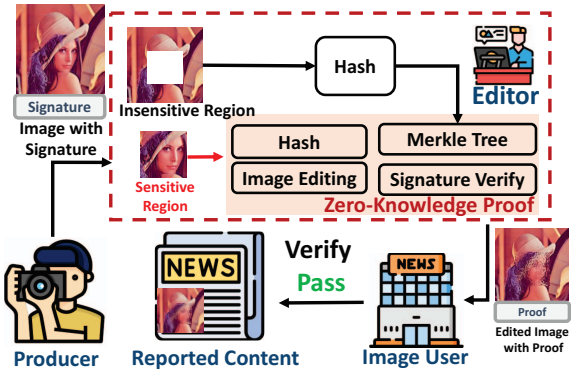

Figure 1: Overview pipeline of our system.

Active image authentication methods can validate the integrity and the editing history of an image by attaching meta-data, such as digital signatures. Traditional digital signature approaches [14] can not fulfill the requirement for detection modification; thus the authenticity is invalid after authorized editing. New methods such as semi-fragile watermarking [15, 19] and perceptual hashing [9, 20] were developed to extend the image authenticity when the images are edited by a set of authorized operations. However, these approaches only allow a limited set of editing functions, such as rotation for $< 5^\circ$ and cropping for $< 10\%$ of image size.

The leading-edge cryptography [6, 7, 17] tools have advanced the research to a new horizon in supporting more editing functions, where the authentication can be carried out using accumulators, chameleon hashing, and Zero-Knowledge Proof (ZKP) [10]. Due to mathematical limitations, cryptography-based methods either suffer from unacceptably long execution time [17] or only support pre-defined editing functions [6, 7]. These drawbacks and trade-offs make them impractical in applications.

Our key insight. We aim to mitigate the trade-off between the execution time and the flexibility of the authorized editing functions. According to the nature of image editing,

2770-4319/23/$31.00 ©2023 IEEE
DOI 10.1109/MIPR59079.2023.00037

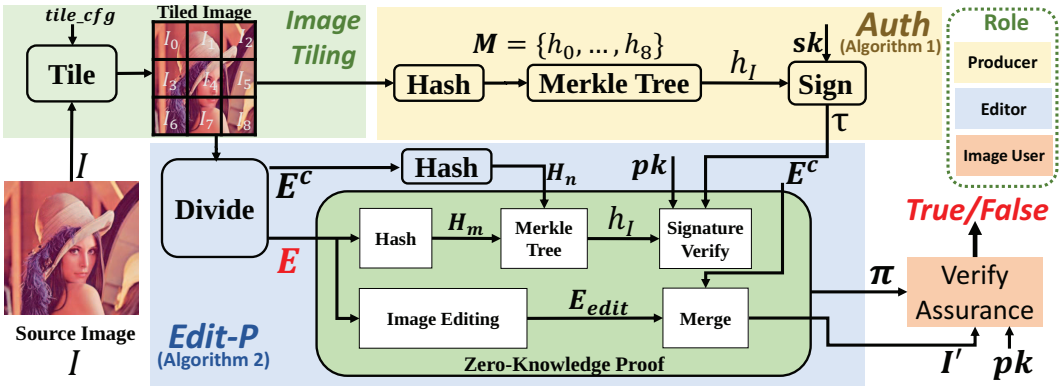

Figure 2: The authentication framework of our Region-Aware Photo Assurance system.

several global editing functions (such as mirroring, rotation, and color adjustments) are usually reversible and content privacy is not an issue. So these global edits can be authenticated using efficient signature-based methods. On the other hand, some local editing functions (such as mosaic, cropping, masking) are usually used to protect small local regions, or *sensitive regions*, where the privacy of the source image should be preserved [18] after editing. As the sensitive regions are usually small (average 17.6% of an image according to the dataset of [18]), we only need to apply sophisticated authentication tools (*e.g.*, ZKP) on the sensitive regions. This can significantly improve the efficiency compared to prior work (*e.g.*, PhotoProof [17] where ZKP is applied to the entire image). See Figures 1 for the overview of our authentication pipeline.

We propose a **region-aware photo assurance system** that can generate image authentication and verify the integrity of an image, for a wide range of editing operations. We separate the image into two parts, namely the *edited* and *non-edited tiles*. For the edited tiles, the idea of digital signature and Zero-knowledge Proof are combined to protect the source image and support flexible editing functions. For the non-edited tiles, the digital signature is suitable to maintain execution efficiency and protect region authenticity. We implemented our system based on the following cryptography tools: (1) Digital Signature [14], (2) a ZKP implementation (zk-SNARKs) [12], (3) Poseidon hash [11] (a ZKP-friendly hash), and (4) Merkle tree [4]. The system supports several editing functions, including mosaic, collage, blur, cropping, and resizing, and has the potential to cover more functions.

We compare our system with several existing image authentication methods to show the proposed system inherits the advantage of the ZKP-based method and largely improves the system's efficiency. Compared with the existing state-of-the-art ZKP-based methods, PhotoProof, our system achieves 15×/60× faster on the KeyGen/Proof operations and 25× lower memory cost. We further investigate the hyper-parameters of our system and show the existence of the optimal setting. Based on the optimal setting, our system achieves significant performance advantages (up to 2700×) on the real-world privacy image dataset, Redaction, compared with PhotoProof. We expect our system can become a practical system for real-world applications.

2 Related Work

We organize the image authentication methods into three categories: Machine Learning (ML) based, Computer Vision (CV) based, and Cryptography (Crypto) based methods.

ML-based methods. As image tampering technologies continue to improve, an amateur in many cases cannot distinguish whether an image has been tampered. Oftentimes, ML algorithms can still detect the tampered regions using meticulous cues. Cozzolino and Verdoliva [8] propose a method called Noiseprint that outputs a heatmap of possible manipulations. Based on the Siamese network that extracts fingerprints of the camera model, Noiseprint significantly suppresses the image content and only highlights the manipulation artifacts. In [2], a Conditional Generative Adversarial Network (cGAN) is used to localize tampered regions in an image. The generator estimates a forgery mask that approximates the true mask, while the discriminator discerns whether the masked region is forged or pristine. Although the ML-based methods show promising results, these methods usually cannot be generalized to cases that are not covered in the training set.

CV-based methods contain two sub-types based on perceptual hashing [9, 20] and semi-fragile watermarking [15, 19]. Both approaches are effective in ensuring the integrity and originality of images. Perceptual hashing uses the perceptual hash to represent an image. Generally, the hash is a local-sensitive hash that similar images usually have a high probability to be categorized into the same hash bucket. This technique is highly suitable for image applications, such as image identification, tamper detection, and content-based image retrieval. Digital watermark methods embed a specially designed pattern into an image to verify image ownership. Because a slight modification usually damages the watermark, it is usually used to detect the modified region in an

Table 1: Comparison between existing authentication methods and our system.

Technique	Type	Primitive	Supported Editing	Negl. Error Probability	Performance	
					Size Overhead	Efficiency
Digital Signature	Active	CP	×	✓	$O(1)$	$O(N)$
[8] [2]	Passive	ML	limited	×	–	–
[15] [19]	Active	SW	limited	×	$O(N)$	$O(N)$
[9] [20]	Active	PH	limited	×	$O(N)$	$O(N)$
VILS [6]	Active	CP	predefined	✓	$O(1)$	$O(K)$
PhotoProof [17]	Active	CP	any	✓	$O(1)$	$O(N\log N)$
Photo Assurance (Ours)	Active	CP	any	✓	$O(1)$	$O(M\log M)$

Primitive can be Cryptography Proofs (CP), Semi-fragile Watermarking (SW), Perceptual Hash (PH), or Machine Learning (ML). N: image size (pixels); M: editing region (pixels); K: number of predefined operations.

image, which is widely used to track copyright infringement and authenticate banknotes. However, the digital watermark cannot distinguish the intention of the modification no matter whether it is malicious or authorized.

Crypto-based methods operate based on cryptography tools for image authentication, thus they are strict and do not tolerate any image modification. The digital signature scheme [14] is commonly used, where the signature is protected by rigorous mathematical theories. An authentication tag is generated by signing the hash values through a pre-generated private key. The image-tag pair is transmitted to the image users for authenticating the image through a pre-generated public key. It is not possible to manipulate an image without invalidating the corresponding tag. However, this naive signature-based approach does not work for the authentication of images after authorized edits. To this end, advanced cryptographic primitives, such as Zero-Knowledge Proof, bilinear-paring, and accumulators are developed.

PhotoProof [17] is an active image authentication method based on the concepts of Zero-knowledge Proof (ZKP) and digital signature. This method achieves security and privacy guarantees for general image editing, where the verification can perform directly on the edited image, while not revealing information of the source image. However, it suffers from unacceptable execution time and memory usage. PhotoProof takes about 360 seconds to generate a proof for an 128×128 image, and the public key used to generate the proof is as large as 2 GB. This disadvantage is due to that the entire image is hashed completely in the ZKP implementation, which is very inefficient. Several approaches [6, 7] trade the editing flexibility for higher system efficiency, in that only a small set of pre-defined editing operations are allowed.

In this paper, we mitigate the trade-off of the ZKP-based methods with a region-aware design. The experiments show that our method is practical for real-world usage.

3 Cryptography Tools

We describe the cryptographic tools used in our system. Generally, a secure **signature scheme** should provide existential unforgeability against adaptive chosen-message attacks, based on the standard definition of [14]. A signature scheme generally contains three algorithms:

- $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$. $\mathsf{KeyGen}(\cdot)$ is a polynomial-time algorithm that takes a security parameter as input and outputs a secret/public key pair $(\mathsf{sk},\mathsf{pk})$.
- $\tau \leftarrow \mathsf{Sign}(\mathsf{sk},x)$. $\mathsf{Sign}(\cdot)$ takes a secret key sk and a message x as input and outputs a tag τ.
- $accept/reject \leftarrow \mathsf{Verify}(\mathsf{pk},\tau,x)$. $\mathsf{Verify}(\cdot)$ takes a public key pk, a message x and a signature τ as input, and returns a verified result of accept or reject (accept implies that the message-tag pair is valid).

Non-interactive Zero Knowledge Proof (ZKP) is a protocol that one party (*prover*) can prove the truthness of a predefined statement to another party (*verifier*), while avoiding conveying any additional information apart from the fact. Requirements that need to be satisfied by any ZKP are:

1. **Completeness**: As long as both the prover and verifier behave honestly, ZKP works for "true proofs".
2. **Soundness**: The prover shall not be able to generate "false proofs" without the knowledge of the witness.
3. **Zero-knowledge**: No knowledge about the prover is revealed to the verifier other than the knowledge of proof being true or false.

We use a standard **Merkle tree** scheme $\mathsf{Merkle} := \{\mathsf{Path},\mathsf{Hash}\}$, where Hash compresses the input into a short digest called *root*. Given the input data, the root, and a location index i, Path outputs a path (of the logarithmic size of the data), proving the consistency of the data at location i. Anyone holding the root can verify efficiently the validity of the consistency claim from Path. A simple Merkle tree scheme follows from any collision-resistant hash functions.

4 Photo Assurance System

Our photo assurance system aims for three key advantages in design: (1) We allow the editor to prove the editing operations on an authenticated sensitive region without revealing it. (2) We allow the user to verify the editor's proof. (3) The system should run efficiently for both proving and verifying schemes. We construct our region-aware image authentication method based on digital signatures, ZKP, and Merkle tree. The digital signature is used to authenticate the source image. We combine the digital signature and ZKP to achieve (1) and (2). To achieve (3), we tile the source image

Algorithm 1: Auth

Input:
Private key sk, original image I, tile config T

Output:
The image-tag pair (I, τ)

Follow the T to tile I into $i_0, i_1, ..., i_{t-1}$
$H = \{Hash(i_k) \mid k \in \{0, 1, ..., t-1\}\}$
$h_I \leftarrow \mathsf{Merkle.Hash}(H)$
$\tau \leftarrow \mathsf{Sig.Sign}(\mathsf{sk}, h_I)$
return (I, τ)

and only use ZKP to authenticate the *edited tiles*[1] because of the long authentication time of ZKP. Furthermore, we use the Merkle tree to jointly combine the hashes of the edited tiles E and non-edited tiles E^c to support the digital signature, which can directly verify the entire image with only a single hash. This design significantly reduces the communication overhead for hash transmission.

Fig. 2 shows our authentication framework. The framework includes three heterogeneous roles, namely, *image producer, image editor, and image user*. In the beginning, the producer tiles the source image I according to a tiling configuration T (with vertical and horizontal tile sizes) and generates a hash set M (which contains a hash per image tile I_i). Then, the producer uses the Merkle tree to concatenate these hashes to generate a single image hash h_I. Image producer signs h_I with private key sk and outputs a signature as the authentication tag τ. The editor builds a Merkle tree in ZKP and merges the hashes of the non-edited tiles H_n and edited tiles H_m again to prove the integrity of the input image using τ. Besides, the editor edits several image tiles E with certain local editing functions to protect the signed sensitive regions and then merges the edited tiles E_{edit} with the non-edited tiles. After verifying the hashes and editing the image with ZKP, the editor generates a ZKP proof π (named **photo assurance**) to jointly authenticate the integrity of the source image and the performed editing function. With the edited image I' and photo assurance π, users can directly verify I' and the editing operations with π by running the ZKP verification algorithm without knowing the source image I.

Detailed Algorithms. Our scheme includes five probabilistic polynomial time algorithms (Setup, KeyGen, Auth, Edit-P, Verify). Setup, KeyGen, and Verify have similar structures as the standard digital signature and ZKP. Due to the space limits, we only detail Auth and Edit-P. Generally, our system uses the following building blocks:

- $\mathsf{Sig} := \{\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}\}$ is a signature scheme.
- $\Pi := \{\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}\}$ is a non-interactive ZKP scheme for general Non-deterministic Polynomial-time (NP) languages.
- $\mathsf{Merkle} := \{\mathsf{Path}, \mathsf{Hash}\}$ is a Merkle tree scheme.

[1]We use the edited tile here because the sensitive regions will be divided by several image tiles and not all sensitive regions will be edited.

Algorithm 2: Edit-P

Input:
Public key pk, the image-tag pair (I, τ), the set of edited tiles E, a tile config T, a ZKP proving and verification key Pub, an editing function f

Output:
The edited image I', the Photo Assurance π

$I = E \cup E^c$
$H_n = \{H(i_n) \mid n \in E^c\}$
The following operations are in ZKP (e.g., Circom).
$H_m = \{H(i_m) \mid m \in E\}$
$H = H_n \cup H_m$
$h_I \leftarrow \mathsf{Merkle.Hash}(H)$
if $\mathsf{Sig.Verify}(\mathsf{pk}, (h_I, \tau)) \neq 1$ **then**
  return reject;
$I' = f(I)$
Take out crs_f from Pub
Run the zero-proof $\pi = \Pi.\mathsf{Prove}(\mathsf{crs}_f, I, f, pk)$
return (I', π)

A photo assurance scheme with respect to the function class $\mathscr{F}$ consists of the following algorithms. In $\mathsf{Setup}(1^\lambda, \mathscr{F})$, for every function $f \in \mathscr{F}$, the algorithm runs $\Pi.\mathsf{crs}_f \leftarrow \Pi.\mathsf{Setup}(1^\lambda, L_f)$ of the following NP language L_f : $\{(\mathsf{pk}, y) : \exists (x, \tau)$ such that $f(x) = y$ and $\mathsf{Sig.Verify}(\mathsf{pk}, x, \tau) = 1\}$. Define $\mathsf{Pub} = \{\Pi.\mathsf{crs}_f\}_{f \in \mathscr{F}}$ and publish Pub as public information. In $\mathsf{KeyGen}(1^\lambda)$, the algorithm sets $(\mathsf{pk}, \mathsf{sk}) := (\mathsf{Sig.pk}, \mathsf{Sig.sk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$.

In **Algorithm 1** (Auth), the original image is tiled into several sub-image I_k according to the tile configuration T. We calculate the hash value h_k of each image tile I_k and run $\mathsf{Merkle.Hash}()$ to obtain the hash h_I of the original image. Finally, the image producer generates the image-tag pair (I, τ) and sends it to the image editor. In **Algorithm 2** (Edit-P), the hash computation of the image tiles of non-editing area is offloaded outside the ZKP. In ZKP, the editor calculates the hash of only the edited image tiles and uses H_n and H_m as the inputs to $\mathsf{Merkle.Hash}(H)$ to calculate the hash h_I of the source image and verify the tag τ of the source image from the image producer. In Verify, given the input of public parameter Pub, public key pk, function $f \in \mathscr{F}$, and a pair of image and assurance (I', π), the image verifier (user) take out crs_f from Pub and runs the $\Pi.\mathsf{Verify}(\mathsf{crs}_f, I', \pi, \mathsf{pk})$ to verify the assurance sent by image editor.

5 Experiment Results

Experimental Settings. We implement our system with two javascript ZKP packages: Circom [16] and snarkjs [12]. Circom is a domain-specific language that defines arithmetic

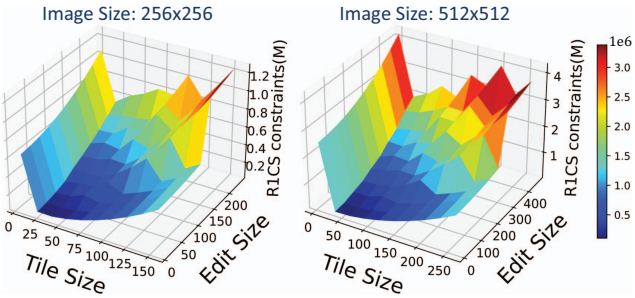

Figure 3: Performance trade-off between the tile and the edited sizes. The number of R1CS constraints is positively correlated to the execution time.

Table 2: Performance Breakdown on Different Image Sizes. R.-A., Sig., and Mkl represent Region-Aware, Signature, and Merkle Tree, respectively.

R.-A.	Size	Edit	Sig.	Hash	Mkl	R1CS (M)
No	32x32	6%	40%	54%	-	0.29
	64x64	8%	12%	79%	-	0.16
	128x128	8%	4%	88%	-	0.57
	256x256	9%	1%	90%	-	2.23
Yes	32x32	7%	52%	18%	23%	0.42
	64x64	11%	19%	28%	43%	0.11
	128x128	20%	9%	51%	20%	0.25
	256x256	26%	3%	65%	6%	0.77

circuits to generate the circuit for ZKP and compiles an editing function to a *Rank-1 Constraint System* (R1CS). The snarkjs package is a JavaScript-based zk-SNARK implementation that carries out the process of generating and verifying the proof. We adopt the Edwards-Curve Digital Signature Algorithm [13] as the digital signature scheme because it is efficient and extensible. To improve hashing efficiency, we adopt a ZKP-friendly hash, POSEIDON [11], for both the digital signature and ZKP schemes.

For the hardware specification, we implement the image editor on a Macbook Air laptop (Core M1, 16GB RAM) with the snarkjs package. We implement the user on a desktop (Intel i7-6700K @4.0GHz CPU, 16 GB RAM), and the Trust-setup server on a workstation (Intel i9-9820X @3.3GHz CPU, 128 GB RAM) with Circom and snarkjs for supporting image editors to generate crs_f. For secure transmission, we adopt the Secure File Transfer Protocol (SFTP). We implemented five common image editing functions including mosaic, collage, color adjustment, cropping, and blur. We omit the function details due to space limitations.

Performance Analysis. The performance bottleneck of our system is the ZKP scheme. We use the number of R1CS constraints as an indicator to evaluate the system performance, as the number and the execution time are highly correlated. We analyze the performance impact of the five ZK modules: *region merging, signature verification, image editing, Merkle tree, and hashing*. The time for region merging can be neglected because the process does not increase the number of R1CS. The time for signature verification is essentially a constant of the security parameter, as in the hash-and-sign paradigm, signature verification of a message can be done by verifying a short hash.

Table 3: Performance for protecting Redaction images. The time represents the proof time. The privacy attributes of No.1-5 are the face, driving license, address, birthday, and credit card, respectively. (*approximate value)

No.	Image Size (pixels)	Edit Region (pixels)	[17]'s Time*	Our Time	Opt.
01	900×674	62,050 (10%)	188 m	58.5 s	180×
02	500×333	754 (0.4%)	51 m	1.75 s	1,740×
03	1,024×683	8,292 (1.2%)	217 m	9.9 s	1,320×
04	2,100×1,394	1,822 (0.06%)	911 m	20.4 s	2,700×
05	1,024×633	17,118 (2.6%)	201 m	17.9 s	660×

To show the effectiveness of region-aware design, we analyzed both non-region-aware and region-aware ZKP circuits. Table 2 shows the performance breakdowns. In the non-region-aware experiment, we can see the difference in time proportion of editing the image, generating a digital signature, performing image hashing, and building the Merkle Tree. The hashing gradually dominates the execution time compared with editing and signature when image size increases. With the region-aware design, the effort for hashing decreases. Then, the total number of R1CS constraints decreases even if the system includes an additional process for building the Murkle tree. The results demonstrate the effectiveness of our region-aware design. Table 4 shows the performance comparison with existing state-of-the-art approaches [6, 7, 17]. The approaches of [6, 7] adopt accumulator and bi-linear pairing methods, however with limited support of editing operations. Thus, this method is less practical, despite that this method runs faster than PhotoProof. In theory, PhotoProof and our system can support an unlimited number of editing functions. Our system achieves 15×/60× faster computation time in KeyGen/Proof and 25× lower Proof size when compared with PhotoProof.

To test the performance of our system on real-world image editing scenarios that involves privacy protection, we conducted experiments on the Redaction dataset [18]. The dataset categorizes the sensitive information of the images into 68 attributes. Overall, the average size of the sensitive region is about 17.6% of the total image pixels. The statistic coincides with our assumption that sensitive regions only take a small portion of the source image. Table 3 reports the execution time of our system on five randomly selected images edited by image mosaic. Compared with PhotoProof [17], our performance improve significantly, ranging from 180× to 2,700× faster. The execution time depends on the size of the source image and the number of edited tiles. This result shows that our region-aware design and system implementation is effective and can greatly speed up

Table 4: Performance comparison of the proposed system with existing methods. The number of possible editing represents the number of supported editing functions. The setting of the editing area in the experiment is 64×64 pixels.

Scheme	Image Size (pixels)	# of Possible Editing	Number of Gates	Time Cost in Seconds (s)			Memory Cost (KB)	
				KeyGen	Proof	Verify	Key Size	Proof Size
[7]	128×128	1000	–	≈ 1	0.33	0.025	13	≤ 2
VILS [6]	128×128	1000	–	≈ 1	0.35	0.018	15	≤ 2
PhotoProof [17]	128×128	Unlimited	12,531,999	≈367	306	0.5	2.6×10^6	2.67
Photo Assurance (Ours)	128×128	Unlimited	200,594	≈ 26	5.45	0.5	1.1×10^5	0.8

photo assurance in real use cases.

Trade-off in our system involves several factors: the image tile size, image editing, Merkle tree construction, and hashing. The time for the construction of the Merkle tree is proportional to the number of leaves (tiles) in the tree. While the tile size increases, the number of leaves decreases, which reduces the time for building the tree. The time for hashing the whole image is positively correlated to image tile size. The tile size also greatly affects the R1CS size for generating the photo assurance π in the ZKP implementation. Therefore, selecting a proper tile size is critical for reducing execution time. Fig. 3 shows the impact of the R1CS size under different edited regions and image sizes. The **best tile size** for a 256×256 image is 20×20, and for 512×512 image it is 45×45, for all edit sizes. Based on the best tile size, we investigate the performance impact caused by the sizes of the edited region and the image. In general, the hash computation is the performance bottleneck in our system, as the hashing time is much larger than the time needed for building the Murkle tree.

6 Conclusion

We present Region-Aware Photo Assurance, an efficient active image authentication system based on ZKP and cryptographic tools. In our system, the image editor can create proofs of editing operations and publish the edited images for open authentication. Any image user can verify the edited image using the proof. We mitigated the trade-off between the flexibility of the editing function and execution efficiency. We demonstrated the practicality of our method with extensive experiments. **Future work** includes extending our framework to additional multimedia types, therefore contributing a step toward trustworthy media content and media forensics.

References

[1] S. Bakhshi, D. Shamma, and E. Gilbert. Faces engage us: Photos with faces attract more likes and comments on Instagram. In *CHI'14*, page 965–974, 2014.

[2] E. Bartusiak, S. Yarlagadda, D. Güera, P. Bestagini, S. Tubaro, F. Zhu, and E. Delp. Splicing detection and localization in satellite imagery using conditional gans. In *MIPR*, pages 91–96, 2019.

[3] S. Battiato, O. Giudice, and A. Paratore. Multimedia forensics: discovering the history of multimedia contents. In *CompSysTech*, pages 5–16, 2016.

[4] G. Becker. Merkle signature schemes, Merkle trees and their cryptanalysis. *Ruhr-Universität Bochum*, 12:19, 2008.

[5] K. Bhagtani, A. Yadav, E. Bartusiak, Z. Xiang, R. Shao, S. Baireddy, and E. Delp. An overview of recent work in media forensics: Methods and threats. *arXiv*, 2022.

[6] H. Chen, X. Huang, J. Ning, F. Zhang, and C. Lin. VILS: A verifiable image licensing system. *IEEE TIFS*, 17:1420–1434, 2022.

[7] H. Chen, X. Huang, W. Wu, and Y. Mu. Efficient and secure image authentication with robustness and versatility. *Science China Information Sciences*, 63(12):1–18, 2020.

[8] D. Cozzolino and L. Verdoliva. Noiseprint: a CNN-based camera model fingerprint. *IEEE TIFS*, 15:144–159, 2019.

[9] R. Davarzani, S. Mozaffari, and K. Yaghmaie. Perceptual image hashing using center-symmetric local binary patterns. *Multimedia Tools and Applications*, 75(8):4639–4667, 2016.

[10] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *Journal of cryptology*, 1(2):77–94, 1988.

[11] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. POSEIDON: A new hash function for zero-knowledge proof systems. In *USENIX Security*, pages 519–535, 2021.

[12] Iden3. SNARKJS: Javascript implementation of zk-SNARKs. https://github.com/iden3/snarkjs, 2022.

[13] S. Josefsson and I. Liusvaara. Edwards-curve digital signature algorithm (EdDSA). Technical report, 2017.

[14] R. Kaur and A. Kaur. Digital signature. In *IEEE ICCS*, pages 295–301, 2012.

[15] C.-Y. Lin and S.-F. Chang. Semifragile watermarking for authenticating jpeg visual content. 3971:140–151, 2000.

[16] J. Muñoz-Tapia, M. Belles, M. Isabel, A. Rubio, and J. Baylina. CIRCOM: A robust and scalable language for building complex zero-knowledge circuits. 2022.

[17] A. Naveh and E. Tromer. PhotoProof: Cryptographic image authentication for any set of permissible transformations. In *IEEE S&P 2023*, pages 255–271, 2016.

[18] T. Orekondy, M. Fritz, and B. Schiele. Connecting pixels to privacy and utility: Automatic redaction of private information in images. In *IEEE CVPR*, pages 8466–8475, 2018.

[19] R. Sun, H. Sun, and T. Yao. A SVD-and quantization based semi-fragile watermarking technique for image authentication. In *ICICSP*, volume 2, pages 1592–1595, 2002.

[20] Z. Tang, X. Zhang, X. Li, and S. Zhang. Robust image hashing with ring partition and invariant vector distance. *IEEE TIFS*, 11(1):200–214, 2015.

[21] L. Wu, F. Morstatter, K. Carley, and H. Liu. Misinformation in social media: Definition, manipulation, and detection. *ACM SIGKDD Explorations Newsletter*, 21(2):80–90, 2019.