Check for updates

Information Hiding - New Applications and Solutions - Research Article

International Journal of Distributed Sensor Networks

International Journal of Distributed Sensor Networks
2021, Vol. 17(11)
© The Author(s) 2021
DOI: 10.1177/15501477211059092
journals.sagepub.com/home/dsn
SAGE

Privacy-preserving inpainting for outsourced image

Fang Cao[1,2], Jiayi Sun[3], Xiangyang Luo[4], Chuan Qin[3] and Ching-Chun Chang[5]

Abstract

In this article, a framework of privacy-preserving inpainting for outsourced image and an encrypted-image inpainting scheme are proposed. Different with conventional image inpainting in plaintext domain, there are two entities, that is, content owner and image restorer, in our framework. Content owner first encrypts his or her damaged image for privacy protection and outsources the encrypted, damaged image to image restorer, who may be a cloud server with powerful computation capability. Image restorer performs inpainting in encrypted domain and sends the inpainted and encrypted image back to content owner or authorized receiver, who can acquire final inpainted result in plaintext domain through decryption. In our encrypted-image inpainting scheme, with the assist of Johnson–Lindenstrauss transform that can preserve Euclidean distance between two vectors before and after encryption, the best-matching block with the smallest distance to current block can be found and utilized for patch filling in Paillier-encrypted image. To eliminate mosaic effect after decryption, weighted mean filtering in encrypted domain is conducted with Paillier homomorphic properties. Experimental results show that our privacy-preserving inpainting framework can be effectively applied in secure cloud computing, and the proposed encrypted-image inpainting scheme achieves comparable visual quality of inpainted results with some typical inpainting schemes in plaintext domain.

Keywords

Privacy preserving, image inpainting, encrypted domain, visual quality

Date received: 25 August 2021; accepted: 8 October 2021

Handling Editor: Yanjiao Chen

Introduction

Image inpainting is also known as image retouching, the idea of which is inherited from ancient technique of manually repairing valuable artworks in an indiscernible way.[1] Inpainting of digital images has found applications in such fields as repairing of historical photographs,[2] filling in or removing the selected region in images,[3] and wiping off visible watermarks.[4] In recent years, some studies have also been investigated to utilize image inpainting technique in deinterlacing,[5] image compression,[6,7] image self-recovery,[8,9] data hiding,[10] and repairing missing blocks of JPEG images due to poor channels.[11] As for traditional image restoration, the regions for restoring generally include both noise and useful information. However, the

[1]College of Information Engineering, Shanghai Maritime University, Shanghai, China
[2]Guangxi Key Lab of Multi-Source Information Mining & Security, Guangxi Normal University, Guilin, China
[3]School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai, China
[4]Zhengzhou Information Science and Technology Institute, Zhengzhou, China
[5]Department of Computer Science, University of Warwick, Coventry, UK

Corresponding author:
Ching-Chun Chang, Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK.
Email: ching-chun.chang@warwickgrad.net

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).

damaged or missing regions to be inpainted often contain no useful information. Hence, the task of inpainting is to produce or create image regions that initially do not exist at all, according to the available information in close neighborhood and some mathematical models. Currently, there are four main categories of image inpainting methods, including interpolation-based methods,[12–15] partial differential equation (PDE)-based methods,[1,2,16–18] exemplar/patch-based methods,[3,19–21] and learning-based methods.[22–25] A brief review of these inpainting methods is given in section "Related works."

Nowadays, with the prosperous development of Internet and mobile network, cloud storage and computing have become more and more prevalent. Due to the limit storage space and computation capability, the users can upload their multimedia data (such as texts, images, audios, and videos) through network, and various kinds of data computing required by users can be outsourced and implemented by cloud server. However, although cloud computing brings promising conveniences, the security issue is also raised at the same time because of the user privacy problem.[26] In other words, users are afraid that their private data will be leaked, or cloud server and the third party may abuse their data. Therefore, a secure and reasonable solution is to encrypt the user data before outsourcing to the cloud for privacy protection, and the cloud server can perform data computing in the encrypted domain. Then, after conducting decryption for the received data from cloud server, the user can obtain the processed data with desirable effects in the plaintext domain. Based on this application scenario and requirements, in recent years, the studies of secure signal processing in encrypted domain have been widely investigated, such as transform,[27] compression,[28] denoising,[29] feature extraction,[30] and data hiding[31,32] for encrypted data.

In this article, we mainly focus on the problem of privacy-preserving inpainting for outsourced images. The aim of this problem is to realize the inpainting for encrypted, damaged images of users on the cloud server without sacrificing user privacy. Currently, although many image processing tools have modules to inpaint images, there are still a lot of manual operations that should be involved. In addition, these tools are often not free, and if we just have only a few images required to be inpainted, it is actually not a cost-effective choice to pay for these software. However, if we need to conduct inpainting for a large number of images, these professional tools often involve many manual operations and complex calculations during the implementation on the user client with limited computation capability and power. If we decide to outsource the inpainting work to the cloud, the privacy of image owner must be considered, which means the images to be inpainted should be encrypted at first. However, after the images are encrypted to prevent information leakage, there are also few related information that can be utilized in the encrypted domain for inpainting. Hence, the challenging and innovative task of outsourcing encrypted, damaged images to the cloud for automatic inpainting without leaking privacy deserves our in-depth investigations and has a lot of application scenarios. To the best of our knowledge, the reported works of inpainting all focused on the plaintext domain, and there have been no published literatures about image inpainting in the encrypted domain.

In this work, we propose a damaged image inpainting scheme based on a new privacy-preserving inpainting framework. The main contributions of the proposed scheme are summarized as follows: (1) The operation of image inpainting in our scheme is completely implemented in the encrypted domain, and image restorer cannot access any information of plaintext-image contents; (2) To calculate priority order and find similar blocks in encrypted image, with the assist of Johnson–Lindenstrauss (JL) transform that preserves Euclidean distance between two vectors before and after encryption, the best-matching, intact block can be found in the source region, and the intact patch is exploited to fill the damaged patch in homomorphic encrypted image; (3) To eliminate the undesirable mosaic effect after decryption, the weighted mean filtering in the encrypted domain is performed using the properties homomorphic cryptosystem; (4) Besides protecting the privacy of image owner, the proposed encrypted-image inpainting scheme can achieve comparable performance of inpainted-image quality with respect to some typical inpainting schemes for plaintext images.

The rest parts of the article are organized as follows. Section "Related works" presents brief reviews of image inpainting and homomorphic encryption. Section "Proposed scheme" describes the detailed procedures of the proposed privacy-preserving inpainting scheme for outsourced image, including image encryption, inpainting for encrypted image, and image decryption. Experimental results and analysis are presented in section "Experimental results and analysis." Section "Conclusion" concludes the article.

Related works

Image inpainting

Generally speaking, current image inpainting methods can be categorized as non-learning-based methods and learning-based methods. Learning-based inpainting methods[22–25] often require a large-scale set of intact images for training with a specific deep neural network and then learn how to generate a reasonable repaired result for damaged region in a given image. Non-

learning-based inpainting methods depend on the characteristic of autocorrelation within natural image, which usually exploit intact information only in current given image beyond its damaged region and use the exploited intact information to repair damaged region seamlessly in different ways, such as interpolation,[12–15] PDE propagation,[1,2,16–18] and patch synthesis.[3,19–21]

Interpolation-based methods. Shih et al.[12] proposed a multi-resolution method using pixel interpolation. In their method, the damaged image for inpainting was segmented into blocks, and the segmentation was conducted until the variances of sub-blocks were smaller than a pre-determined threshold. Damaged pixel was then repaired through the mean value of current sub-block or sub-block of previous level. Another image inpainting scheme with interpolation strategy was proposed in Shih et al.,[13] which evaluated neighboring information of each pixel to be inpainted and decided the size of reference window that can be exploited to calculate an interpolated color. However, these methods may lead to blurring effects when damaged pixels were close to image edges.

PDE-based methods. This type of methods was motivated by intensive works on the use of variational and PDE methods in image processing. Bertalmio et al.[2] established an inpainting mathematical model by borrowing ideas from classic fluid dynamics. By iteratively solving the numerical representation of a PDE, intact information of neighboring areas can be smoothly propagated into damaged region along isophote direction. Guided by the connectivity principle of human visual perception, Chan and Shen[16] proposed a non-texture image inpainting method with a third-order PDE, which essentially was an anisotropic diffusion process based on the total variation (TV) model.[17] To meet the requirements of human vision, diffusion intensity was related to the curvature. Even if the remaining objects were disconnected far apart by damaged region, this method can still output acceptable inpainted result. However, the PDE-based methods cannot deal with the large-area inpainting very well and often introduced heavy computation complexity due to high-order mathematic models.

Exemplar/patch-based methods. This type of methods employed the strategy of texture synthesis[33] to repair larger damaged region. Criminisi et al.[20] presented an exemplar-based image inpainting method, which can be used for both region filling and object removal. This method implemented the task of patch synthesis through a best-first filling mechanism, and the value of priority for each patch depended on the percentage of intact pixels and the angle between the isophote and contour normal on the center pixel. After locating the patch with the maximum priority in damaged region, the most similar patch was selected for substitution from the intact region in the image, and then the priority values should be updated to continue above process repeatedly. Inpainting procedure was terminated till all damaged patches were repaired. This method can achieve better inpainted results for larger damaged region than the PDE-based methods, although it may leave some artificial seams between the patches.

Learning-based methods. In recent years, a lot of works have applied the convolutional neural network (CNN) to image inpainting, such as Iizuka et al.[22] and Pathak et al.[25] In these tasks, training samples were utilized to train the deep CNN so that they can estimate the pixel strength of the damaged region in the input image.[24] Benefiting from large-scale training data, these learning-based methods can produce semantic specious inpainting results. However, the existing CNN-based inpainting methods usually completed damaged region by propagating convolution feature to the fully connected layer, which sometimes made the inpainted results lack fine texture details with blurring. Another powerful family of learning-based inpainting methods with deep generative model[23] has also been proposed through introducing the adversarial loss to improve the visual quality of the inpainted results. Generally speaking, learning-based methods can achieve superior global semantic structures of inpainted results compared with non-learning-based methods, while non-learning-based methods can obtain fine local textures of inpainted results with much less computational complexity.

In this work, we mainly focus on the non-learning-based image inpainting methods, which have efficient and clear algorithms to follow and are more possible to be implemented in the encrypted domain with privacy-preserving capability than the learning-based methods.

Homomorphic encryption

To achieve computing in the encrypted domain, appropriate encryption methods should be first addressed. Homomorphic encryption generally includes partially homomorphic encryption (PHE), fully homomorphic encryption (FHE), and somewhat homomorphic encryption (SHE). PHE allows either addition or multiplication operations. Rivest et al.[34] proposed a homomorphic encryption algorithm, which can maintain some algebraic relationships between the plaintext and the ciphertext. These relationships can be exploited to realize the computation for encrypted data effectively. Afterward, several probabilistic public-key

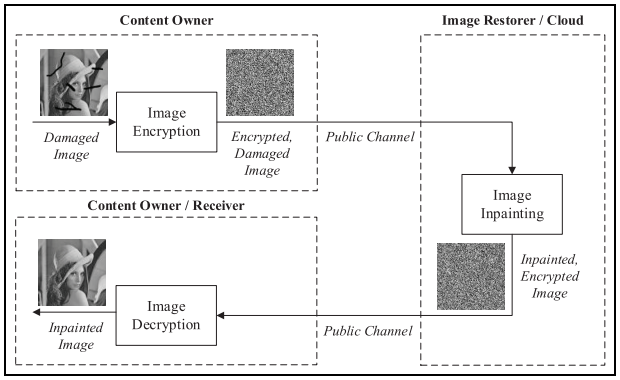

Figure 1. Framework of privacy-preserving inpainting for outsourced image.

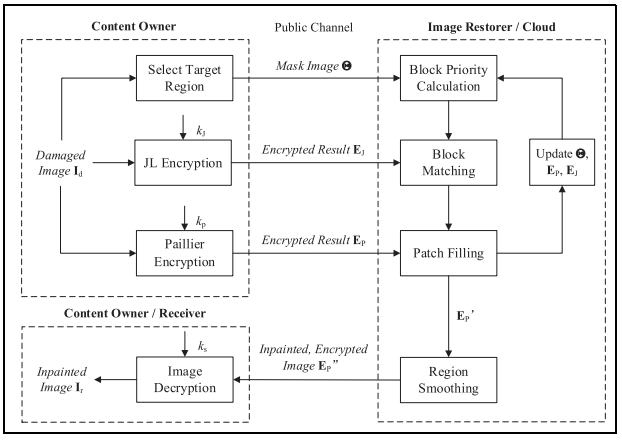

Figure 2. Flowchart of the proposed encrypted-image inpainting scheme.

cryptosystems were presented, such as ElGamal[35] cryptosystem, Paillier[36] cryptosystem, and Damgård and Jurik[37] cryptosystem, and these cryptographic algorithms only had one homomorphic property, that is, addition or multiplication. For example, Paillier cryptosystem only had the addition homomorphism, which means that the addition of two plaintexts can be achieved through performing some operations on the two corresponding ciphertexts. FHE allows arbitrary number of addition and multiplication operations.[38] In other words, FHE can realize the homomorphisms of addition and multiplication simultaneously, and theoretically speaking, it can solve any privacy-preserving computation problems. SHE allows addition and multiplication operations, but the number of operations is limited.[39]

Proposed scheme

In this section, we first present a new privacy-preserving inpainting framework for outsourced image, which can be effectively applied in the environment of cloud computing. Generally speaking, there are two entities in our framework, that is, content owner and image restorer, see Figure 1. Content owner, who has no professional inpainting capability, wants to have his or her damaged image repaired without disclosing image contents for privacy preserving. Hence, the ideal framework for this application scenario is that the content owner first encrypts the damaged image for inpainting and outsources the encrypted image to the image restorer, who may be a cloud server with powerful computation capability. Then, the restorer can conduct the inpainting operation for the damaged image effectively in the encrypted domain and sends the inpainted, encrypted image back to the content owner or the legal receiver, who can directly obtain the inpainted result in plaintext domain through image decryption.

To achieve the above described functions in the framework, a specific encrypted-image inpainting scheme is also proposed. Content owner first chooses a target region $\mathbf{\Omega}$ to be removed or filled in the damaged image $\mathbf{I}_d$ and labels the position of the target region through a binary mask image $\mathbf{\Theta}$, in which the pixels with the values of 0 and 1 denote intact area and damaged area at the corresponding coordinates in the damaged image $\mathbf{I}_d$, respectively. Then, the damaged image $\mathbf{I}_d$ is encrypted by Paillier homomorphic cryptosystem and JL transform to produce two encrypted results, that is, $\mathbf{E}_P$ and $\mathbf{E}_J$. The content owner sends $\mathbf{E}_P$, $\mathbf{E}_J$, and $\mathbf{\Theta}$ to image restorer for inpainting through the public channel. After receiving $\mathbf{E}_P$, $\mathbf{E}_J$, and $\mathbf{\Theta}$, the image restorer can perform image inpainting in the encrypted domain by utilizing these three components. In detail, the priority is first calculated for each pixel belonging to the target region according to the mask image $\mathbf{\Theta}$. With the assist of $\mathbf{E}_J$, the encrypted, damaged image $\mathbf{E}_P$ can be inpainted as $\mathbf{E}_P'$ through the mechanism of sample patch filling in the priority order. Then, the target region of $\mathbf{E}_P'$ is further smoothed with the weighted mean filtering to produce the final inpainted result $\mathbf{E}_P''$ in the encrypted domain. The content owner can decrypt $\mathbf{E}_P''$ to obtain the inpainted image $\mathbf{I}_r$ in the plaintext domain. The flowchart of our scheme is illustrated in Figure 2. Details are described in the following subsections.

Image encryption

To protect the privacy of content owner, the damaged image $\mathbf{I}_d$ to be outsourced for inpainting is first encrypted as two results through the cryptographic techniques, that is, Paillier homomorphic cryptosystem and JL transform. In detail, one encrypted result $\mathbf{E}_P$ is obtained by Paillier homomorphic cryptosystem, which enables the filling or removing for the target region $\mathbf{\Omega}$ with sample patches and the performing of the weighted

mean filtering; the other encrypted result $\mathbf{E}_J$ is obtained by JL transform, which enables the operation of block matching. In addition, a mask image $\mathbf{\Theta}$ is also generated and used to label the position of target region $\mathbf{\Omega}$ for inpainting.

Encryption with Paillier homomorphic cryptosystem. The content owner first encrypts each pixel of the damaged image $\mathbf{I}_d$ sized $H \times W$ in the plaintext domain through homomorphic cryptosystem and obtains the encrypted result of each pixel by equation (1)

$$\rho(i,j) = \phi\,[m(i,j),\ \kappa_p\,] \tag{1}$$

where φ denotes the encryption function based on homomorphic cryptosystem, $m(i, j)$ denotes the value of the pixel in $\mathbf{I}_d$ at the coordinate (i, j), κ_p is the public key of content owner, and $\rho(i, j)$ denotes the encrypted result for the pixel $m(i, j)$. Details of the calculation in equation (1) are given as follows.

In our scheme, Paillier[36] homomorphic encryption belonging to the RSA-based probabilistic public-key cryptosystem is utilized. In Paillier homomorphic encryption, the content owner first chooses two large primes p and q randomly and calculates $n = p{\cdot}q$ and $\lambda = \mathrm{lcm}(p-1, q-1)$, where $\mathrm{lcm}(\cdot)$ denotes the function of the least common multiple. Then, an integer $g \in \mathbf{Z}_{n2}^{*}$ is selected, which should satisfy $\gcd\{[(g^{\lambda} \bmod n^2) - 1]/n,\ n\} = 1$. The function $\gcd(\cdot)$ is used to return the greatest common divisor. Thus, the public key κ_p and the private key κ_s of content owner can be acquired

$$\kappa_p = (n,\ g) \tag{2}$$

$$\kappa_s = (p,\ q,\ \lambda) \tag{3}$$

After generating the public key and the private key, the encryption for each pixel in $\mathbf{I}_d$ can be conducted with the public key κ_p, see equation (4)

$$\rho(i,j) = g^{m(i,j)} \cdot r^n(i,j) \mod n^2 \tag{4}$$

where each pixel value for encryption, that is, the plaintext $m(i, j) \in \mathbf{Z}_n$, and $r(i, j) \in \mathbf{Z}_n^{*}$ that is an n coprime, random number (blinding factor). It should be noted that due to the randomness of $r(i, j)$, the pixels with the same value and different coordinates may have different encrypted results. After all $H \times W$ pixels $m(i, j)$ in the damaged image $\mathbf{I}_d$ are encrypted with equation (4), the content owner collects all corresponding encrypted pixels $\rho(i, j)$ to form the encrypted image $\mathbf{E}_P$.

Actually, besides Paillier homomorphic encryption, the encryption method to generate $\mathbf{E}_P$ in our scheme can also be compatible and replaced with some other additive homomorphic cryptosystems effectively. In addition, the Ring learning with errors (RLWE)-based SHE cryptosystem may also be utilized to further improve the computational efficiency.

Encryption with JL transform. To achieve the operation of block matching in our encrypted-image inpainting, the JL transform, which can not only reduce dimension but also retain Euclidean distance,[40] is also utilized for image encryption. The property of JL transform is based on the following lemma.

Lemma 1. For an arbitrary set $\mathbf{V}$ with N vectors belonging to $\mathbf{R}^d$, given $0 < \varepsilon < 1$ and $k \geqslant \log_2 N/\varepsilon^2$, there exists a linear mapping Γ that can transform each vector of $\mathbf{V}$ from $\mathbf{R}^d$ into $\mathbf{R}^k$, and all pairwise distances after and before mapping are within $1 \pm \varepsilon$ factor. That is to say, for any two vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ in $\mathbf{V}$, the following relationship of the inequality satisfies

$$\begin{aligned}&(1-\varepsilon)\cdot \|\,\boldsymbol{\alpha}-\boldsymbol{\beta}\,\|_2^2\\ &\leqslant \|\,\Gamma(\boldsymbol{\alpha})-\Gamma(\boldsymbol{\beta})\,\|_2^2 \leqslant (1+\varepsilon)\cdot \|\,\boldsymbol{\alpha}-\boldsymbol{\beta}\,\|_2^2\end{aligned} \tag{5}$$

where $\boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbf{R}^d$, and $\Gamma(\boldsymbol{\alpha}), \Gamma(\boldsymbol{\beta}) \in \mathbf{R}^k$.

Through JL transform, a d-dimension vector can be converted to a k-dimension vector, and the Euclidean distance of two d-dimension vectors is approximate to that of the corresponding two k-dimension vectors after the transform. In Kenthapadi et al.,[41] they analyzed the security of JL transform and proved that JL transform can be utilized for data encryption effectively. In our scheme, a random matrix $\Re$ sized $k \times d$ ($\Re \in \mathbf{R}^{k \times d}$), associated with the secret key κ_J, is used as the linear mapping Γ in equation (5), that is, $\Gamma(\boldsymbol{\alpha}) = \Re{\cdot}\boldsymbol{\alpha}$. Note that, the transform from $\boldsymbol{\alpha}$ to $\Gamma(\boldsymbol{\alpha})$ by $\Re$ is an irreversible process when $k < d$.

When encrypting the damaged image $\mathbf{I}_d$ by JL transform, for each pixel $m(i, j)$ in $\mathbf{I}_d$, its neighboring $s-1$ pixels are collected to form an s-pixels local pattern with the pixel $m(i, j)$ at the center. For the pixels on the image border, incomplete pixels in the local pattern can be created with surrounding pixels. Reshape the s-pixels in the local pattern with the center of $m(i, j)$ into an s-dimension vector, which is denoted as $\mathbf{N}_{i,\ j}$. Then, according to the secret key κ_J of JL transform, a random matrix $\Re$ sized $k \times s$ and a random k-dimension Gaussian noise vector σ with zero mean are generated and utilized for encryption

$$\mathbf{L}_{i,j} = \Re \times \mathbf{N}_{i,j} + \boldsymbol{\sigma} \tag{6}$$

where $\mathbf{L}_{i,\ j}$ is a k-dimension vector and denotes the encrypted result for the pixel $m(i, j)$. Note that, different with the Paillier encryption in section "Encryption with Paillier homomorphic cryptosystem," the encrypted result of each pixel by JL transform is a vector instead of a single value. After all $H \times W$ pixels $m(i, j)$ in $\mathbf{I}_d$

are encrypted through equation (6), the content owner collects all corresponding encrypted results $\mathbf{L}_{i,j}$ to form the encrypted image $\mathbf{E}_J$.

Generation of mask image. Similar with the conventional image inpainting in the plaintext domain, the location of the image region to be inpainted should be provided to the restorer, who does not compromise the security of the scheme. Therefore, a binary mask image $\mathbf{\Theta}$ for locating the target region $\mathbf{\Omega}$ in the damaged image $\mathbf{I}_d$ ($\mathbf{E}_P$) is produced by the content owner

$$\theta(i,j) = \begin{cases} 1, & (i,j) \in \Omega, \\ 0, & (i,j) \notin \Omega, \end{cases} \tag{7}$$

where $\theta(i, j)$ is the value of the pixel at the coordinate (i, j) in the mask image $\mathbf{\Theta}$, and $\mathbf{\Theta}$ is also with the size of $H \times W$, which is the same as $\mathbf{I}_d$ and $\mathbf{E}_P$.

After the above operations, including Paillier encryption, JL transform, and mask generation, the content owner sends the obtained results $\mathbf{E}_P$, $\mathbf{E}_J$, and $\mathbf{\Theta}$ to the image restorer for inpainting in the encrypted domain.

Inpainting for encrypted image

After receiving the two encrypted images $\mathbf{E}_P$, $\mathbf{E}_J$, and the mask image $\mathbf{\Theta}$, image restorer conducts patch filling for image inpainting through the exemplar-based texture synthesis in the encrypted domain, see Figure 3. The target region $\mathbf{\Omega}$ in $\mathbf{E}_P$ is first located according to the positions of non-zeros in $\mathbf{\Theta}$. To determine the order of inpainting, for each pixel c in Ω, a block $\mathbf{B}_c$ sized $l \times l$ with c at the center is exploited to calculate the priority. Among all pixels in Ω, the block $\mathbf{B}_{c^*}$ with the highest priority is chosen as the first for repairing.

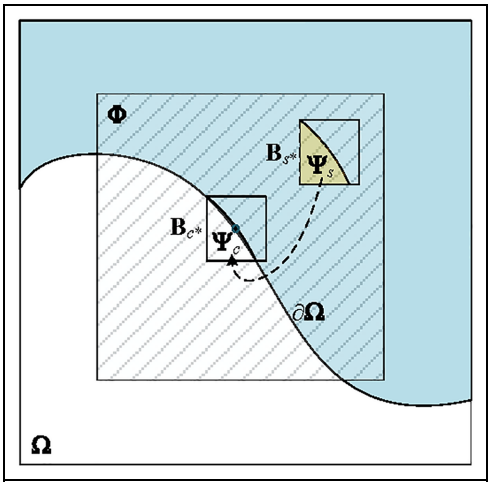

Figure 3. Illustration of patch filling by exemplar-based texture synthesis in encrypted domain.

A source region $\mathbf{\Phi}$ that contains sufficient, intact pixels is defined for the candidate block searching. Since JL transform preserves Euclidean distance, thus, with the assist of $\mathbf{E}_J$, the best-matching block $\mathbf{B}_{s^*}$ with respect to $\mathbf{B}_{c^*}$ can be found in $\mathbf{\Phi}$, and then, in the encrypted image $\mathbf{E}_P$, the damaged patch of $\mathbf{B}_{c^*}$ can be filled with the corresponding patch of $\mathbf{B}_{s^*}$. After that, the values in the mask image $\mathbf{\Theta}$ corresponding to the pixels in the filled patch are marked as zeros, and the target region $\mathbf{\Omega}$, the source region $\mathbf{\Phi}$, and the encrypted image $\mathbf{E}_J$ are updated. The block priorities for the updated Ω are recalculated, and the above procedure is iterated until the target region $\mathbf{\Omega}$ becomes empty. To further smooth the inpainted region, the intermediate result $\mathbf{E}_P'$ after patch filling is performed with the weighted mean filtering in encrypted domain based on Paillier homomorphic properties to produce the final inpainted result $\mathbf{E}_P''$. Details are described as follows.

Block priority calculation. To inpaint the target region $\mathbf{\Omega}$ of the encrypted image $\mathbf{E}_P$, the order of inpainting for all the pixels in $\mathbf{\Omega}$ should be first determined, which is based on the priority of the block with each pixel of $\mathbf{\Omega}$ at the center. Since image restorer does not know the image contents and pixel values from $\mathbf{E}_P$ and $\mathbf{E}_J$ in our encrypted-image inpainting scheme, hence, only the location information of intact and damaged pixels from $\mathbf{\Theta}$ can be utilized for block priority calculation. Obviously, the original intact pixels can be used as references during block matching. Therefore, rather than inpainting from inner to outer, it is more appropriate to perform inpainting for $\mathbf{\Omega}$ from the outer region to the inner region because the blocks on the boundary of $\mathbf{\Omega}$, that is, $\partial\Omega$, include some intact pixels. Thus, the useful information of intact pixels can be effectively used and propagated into target region $\mathbf{\Omega}$, which can lead to a more reasonable inpainted result.

Based on the above analysis, in our scheme, two conditions are considered during the calculation of the block priority: (1) the center pixels of the blocks with higher priorities should be on the boundary of $\mathbf{\Omega}$, that is, $\partial\Omega$; and (2) the blocks with higher priorities should have more intact or inpainted pixels. These two conditions can guarantee a best-first filling mechanism during inpainting and are biased toward those blocks that are on the continuous, strong edges and surrounded by high-confidence pixels. In the target region $\mathbf{\Omega}$ of $\mathbf{E}_P$, the block priority O_c corresponding to the block $\mathbf{B}_c$ with the pixel c at the center can be calculated by

$$O_c = \begin{cases} \frac{\Im(\mathbf{B}_c)}{l^2}, & c \in \partial\Omega, \\ 0, & c \notin \partial\Omega, \end{cases} \tag{8}$$

where the function $\Im(\mathbf{B}_c)$ returns the number of intact pixels in the $l \times l$ block $\mathbf{B}_c$ with the pixel c at the center, and the value of the block priority O_c belongs to [0,

1]. Actually, the number of intact pixels in the block $\mathbf{B}_c$ of the encrypted image $\mathbf{E_P}$ can be counted by the number of zeros in the corresponding block of the mask image $\mathbf{\Theta}$. After the block priorities O_c for all pixels c on $\partial\Omega$ are obtained, the block $\mathbf{B}_{c^*}$ with the highest priority is chosen for candidate block matching, where

$$c^* = \arg\max_c (O_c) \tag{9}$$

It should be noted that, the size $l \times l$ of the blocks for priority calculation and further block matching may have influence on the performance of inpainting. Too small size of the blocks will weaken texture features in the inpainted results. However, if the block size is too large, inpainted results will appear the serious mosaic effect. In addition, the size $r \times r$ of the source region $\mathbf{\Phi}$ is related to inpainting quality and computation complexity. Therefore, the selection of appropriate sizes of block and source region is important, which is discussed in section "Experimental results and analysis."

Block matching and patch filling. After obtaining the block $\mathbf{B}_{c^*}$ with the highest priority in current target region Ω of $\mathbf{E_P}$, the block matching between $\mathbf{B}_{c^*}$ and all candidate blocks in source region $\mathbf{\Phi}$ is conducted with the assist of $\mathbf{E_J}$, and then the corresponding patch $\mathbf{\Psi}_{s^*}$ in the best-matching candidate block $\mathbf{B}_{s^*}$ is utilized to fill the damaged patch $\mathbf{\Psi}_{c^*}$ in $\mathbf{B}_{c^*}$.

The operation of block matching is based on the property of Euclidean distance preserving for JL transform. To conduct block matching and patch filling, the distance between the block $\mathbf{B}_{c^*}$ and each candidate block $\mathbf{B}_s$ from $\mathbf{\Phi}$ is calculated according to $\mathbf{E_J}$

$$dist(\mathbf{B}_{c^*}, \mathbf{B}_s) = \frac{\sum_{(i,j)\in\mathbf{B}_{c^*}} D\left[\mathbf{E_J}(i,j), \mathbf{E_J}(i+x, j+y)\right]}{l^2} \quad \text{subject to } \mathbf{B}_s \in \mathbf{\Phi}, \tag{10}$$

where (x, y) is the coordinate displacement of $\mathbf{B}_s$ with respect to $\mathbf{B}_{c^*}$, s can be considered as the center pixel in $\mathbf{B}_s$, and the function D for the calculation of the distance between the two points of k-dimension vectors in $\mathbf{E_J}$ can be found in equation (11)

$$D\left[\mathbf{E_J}(i_1,j_1), \mathbf{E_J}(i_2,j_2)\right] = \sqrt{\left\| \mathbf{L}_{i_1,j_1} - \mathbf{L}_{i_2,j_2} \right\|_2^2 - 2k \cdot \| \boldsymbol{\sigma} \|_2^2} \tag{11}$$

where $\mathbf{L}_{i1,\ j1}$ and $\mathbf{L}_{i2,\ j2}$ denote two points of k-dimension vectors in $\mathbf{E_J}$, that is, $\mathbf{E_J}(i_1, j_1)$ and $\mathbf{E_J}(i_2, j_2)$, which are also the two encrypted results based on JL transform for the plaintext pixels $m(i_1, j_1)$ and $m(i_2, j_2)$, respectively (refer to section "Encryption with JL transform"). Because the Euclidean distance between two vectors in JL-encrypted domain is approximate to that in the plaintext domain, thus, the candidate block $\mathbf{B}_s$ from $\mathbf{\Phi}$ with the smallest distance to the block $\mathbf{B}_{c^*}$ is selected as the best-matching block $\mathbf{B}_{s^*}$, where

$$s^* = \arg\min_s [dist(\mathbf{B}_{c^*}, \mathbf{B}_s)] \tag{12}$$

Through obtaining the best-matching block $\mathbf{B}_{s^*}$ in the encrypted image $\mathbf{E_P}$, the damaged patch $\mathbf{\Psi}_{c^*}$ of $\mathbf{B}_{c^*}$ can be filled with the corresponding patch $\mathbf{\Psi}_{s^*}$ of $\mathbf{B}_{s^*}$. In other words, all the encrypted, damaged pixels in $\mathbf{B}_{c^*}$ are replaced by the encrypted, intact pixels at the same locations in $\mathbf{B}_{s^*}$, see equation (13)

$$\mathbf{E_P}'(i,j) = \mathbf{E_P}(i + x^*, j + y^*), \quad \text{for } \forall(i,j) \in \mathbf{B}_{c^*} \text{ and } \theta(i,j) = 1 \tag{13}$$

where (x^*, y^*) is the coordinate displacement of $\mathbf{B}_{s^*}$ with respect to $\mathbf{B}_{c^*}$, and $\mathbf{E_P}'(i, j)$ is the filled result for the encrypted, damaged pixel $\mathbf{E_P}(i, j)$. Besides the patch filling for the block $\mathbf{B}_{c^*}$ in $\mathbf{E_P}$, the encrypted image $\mathbf{E_J}$ should also be repaired through the similar process

$$\mathbf{L}'_{i,j} = \mathbf{L}_{i+x^*, j+y^*}, \quad \text{for } \forall(i,j) \in \mathbf{B}_{c^*} \text{ and } \theta(i,j) = 1 \tag{14}$$

where $\mathbf{L}'_{i,\ j}$ denotes the filled result for the encrypted, damaged vector $\mathbf{L}_{i,\ j}$, that is, $\mathbf{E_J}(i, j)$, in $\mathbf{E_J}$. After the patch filling in $\mathbf{E_P}$ and $\mathbf{E_J}$, the mask image $\mathbf{\Theta}$ is required for updating

$$\theta'(i,j) = \begin{cases} 1, & (i,j) \in \Omega - \mathbf{B}_{c^*}, \\ 0, & (i,j) \notin \Omega - \mathbf{B}_{c^*}, \end{cases} \tag{15}$$

where $\theta'(i, j)$ is the updated value of the pixel at the coordinate (i, j) in the new mask image $\mathbf{\Theta}'$. Equation (15) means that the damaged patch in $\mathbf{B}_{c^*}$ is repaired and can be marked as intact. Note that, the target region $\mathbf{\Omega}$ and the source region $\mathbf{\Phi}$ should also be updated at the same time with $\mathbf{\Theta}$, that is, the new $\mathbf{\Omega}'$ is updated with $\mathbf{\Omega} - \mathbf{B}_{c^*}$ and the new $\mathbf{\Phi}'$ is updated with $\mathbf{\Phi} + \mathbf{B}_{c^*}$.

According to the above described steps, the operations of block priority calculation, block matching, and patch filling are iteratively implemented for the updated target region $\mathbf{\Omega}'$. Until the target region becomes empty, the iteration procedure is terminated and generates an inpainted result for the encrypted image $\mathbf{E_P}$, that is, $\mathbf{E_P}'$.

Region smoothing with weighted mean filtering. Because the process of patch filling in section "Block matching and patch filling" is performed in a blockwise manner, thus, undesirable mosaic effect may appear in the target region $\mathbf{\Omega}$ if the inpainted result $\mathbf{E_P}'$ is directly decrypted. To eliminate the mosaic effect, the post-

processing should be conducted for $\mathbf{E}_P'$ in the encrypted domain. Thereby, after image decryption by content owner or legal receiver, an inpainted image in the plaintext domain with satisfactory quality can be produced.

In our scheme, after patch filling, image restorer conducts region smoothing for the target region $\mathbf{\Omega}$ in the intermediate result $\mathbf{E}_P'$ to achieve the equivalent function of weighted mean filtering in the plaintext domain. The operation of region smoothing in the encrypted domain is realized based on homomorphic properties of Paillier cryptosystem. The utilized homomorphic properties of addition are described as follows.

Assume $m(i_1, j_1)$ and $m(i_2, j_2)$ as the values of two pixels at the coordinates (i_1, j_1) and (i_2, j_2) in plaintext domain, respectively. Denote $\rho(i_1, j_1)$ and $\rho(i_2, j_2)$ as the encrypted results of $m(i_1, j_1)$ and $m(i_2, j_2)$ based on Paillier homomorphic encryption (refer to section "Encryption with Paillier homomorphic cryptosystem"), see equation (16)

$$\rho(i_\nu, j_\nu) = g^{m(i_\nu, j_\nu)} \cdot r^n(i_\nu, j_\nu) \mod n^2, \quad \nu = 1, 2 \tag{16}$$

Then, the following three equations (17)–(19) can be acquired, where k is an integer

$$\begin{aligned} \rho(i_1, j_1) \cdot g^k \bmod n^2 &= g^{m(i_1, j_1)} \cdot r^n(i_1, j_1) \cdot g^k \bmod n^2 \\ &= g^{m(i_1, j_1) + k} \cdot r^n(i_1, j_1) \bmod n^2 \end{aligned} \tag{17}$$

$$\begin{aligned} &\rho(i_1, j_1) \cdot \rho(i_2, j_2) \bmod n^2 \\ &= g^{m(i_1, j_1)} \cdot r^n(i_1, j_1) \cdot g^{m(i_2, j_2)} \cdot r^n(i_2, j_2) \bmod n^2 \\ &= g^{m(i_1, j_1) + m(i_2, j_2)} [r(i_1, j_1)\, r(i_2, j_2)]^n \bmod n^2 \end{aligned} \tag{18}$$

$$\begin{aligned} \rho^k(i_1, j_1) \bmod n^2 &= [g^{m(i_1, j_1)} \cdot r^n(i_1, j_1)]^k \bmod n^2 \\ &= g^{k \cdot m(i_1, j_1)} r^{k \cdot n}(i_1, j_1) \bmod n^2 \end{aligned} \tag{19}$$

It can be observed from equations (17)–(19) that, under Paillier cryptosystem, $\rho(i_1, j_1) \cdot g^k \bmod n^2$ is a valid encrypted result for $m(i_1, j_1) + k$, $\rho(i_1, j_1) \cdot \rho(i_2, j_2) \bmod n^2$ is a valid encrypted result for $m(i_1, j_1) + m(i_2, j_2)$, and $\rho^k(i_1, j_1) \bmod n^2$ is a valid encrypted result for $k \cdot m(i_1, j_1)$, respectively.

As for the weighted mean filtering in the plaintext domain, the current pixel $m(i, j)$ in target region $\mathbf{\Omega}$ for processing should be substituted with the weighted mean value of $m(i, j)$ itself and its neighboring pixels

$$\begin{aligned} m'(i,j) &= \mathbf{W}_{i,j} \cdot \mathbf{M}_{i,j}^{\mathrm{T}} \\ &= [w_{i-1,j},\ w_{i+1,j},\ w_{i,j},\ w_{i,j-1},\ w_{i,j+1}] \cdot \\ &\quad [m(i-1,j),\ m(i+1,j),\ m(i,j),\ m(i,j-1),\ m(i,j+1)]^{\mathrm{T}} \\ &= w_{i-1,j} \cdot m(i-1,j) + w_{i+1,j} \cdot m(i+1,j) + w_{i,j} \cdot m(i,j) + \\ &\quad w_{i,j-1} \cdot m(i,j-1) + w_{i,j+1} \cdot m(i,j+1), \end{aligned} \tag{20}$$

where $\mathbf{M}_{i,j}$ denotes the pixel vector consisting of current pixel $m(i, j)$ and its four neighborhood, $\mathbf{W}_{i,j}$ is the weight vector for $\mathbf{M}_{i,j}$, and $m'(i, j)$ is the result of weighted mean filtering for $m(i, j)$. In our scheme, the components of the weight vector $\mathbf{W}_{i,j}$ can be set as: $w_{i-1,j} = w_{i+1,j} = w_{i,j-1} = w_{i,j+1} = 1/(4 + \omega)$ and $w_{i,j} = \omega/(4 + \omega)$, where ω is a small positive integer. To realize weighted mean filtering in encrypted domain based on Paillier homomorphic properties of equations (17) and (19), the components of the weight vector $\mathbf{W}_{i,j}$ should be modified as integers

$$\mathbf{W}'_{i,j} = (4 + \omega) \cdot \mathbf{W}_{i,j} \tag{21}$$

where $\mathbf{W}'_{i,j} = [w'_{i-1,j}, w'_{i+1,j}, w'_{i,j}, w'_{i,j-1}, w'_{i,j+1}] = [1, 1, \omega, 1, 1]$. Thus, for the intermediate result $\mathbf{E}_P'$ after patch filling, according to the homomorphic properties of addition in equations (17)–(19), image restorer can perform the following operation to realize weighted mean filtering in the encrypted domain

$$\begin{aligned} \rho''(i,j) &= \rho'(i-1,j)^{w'_{i-1,j}} \cdot \rho'(i+1,j)^{w'_{i+1,j}} \cdot \\ &\quad \rho'(i,j)^{w'_{i,j}} \cdot \rho'(i,j-1)^{w'_{i,j-1}} \cdot \rho'(i,j+1)^{w'_{i,j+1}} \\ &= \rho'(i-1,j) \cdot \rho'(i+1,j) \cdot \rho'(i,j)^{\omega} \cdot \\ &\quad \rho'(i,j-1) \cdot \rho'(i,j+1), \\ &\quad \text{for } \forall (i,j) \in \mathbf{\Omega}, \end{aligned} \tag{22}$$

where $\rho'(i, j)$ denotes the value of the encrypted pixel at coordinate (i, j) in $\mathbf{E}_P'$, and $\rho''(i, j)$ is the result of weighted mean filtering for $\rho'(i, j)$ in Paillier-encrypted domain.

After all pixels $\rho'(i, j)$ in $\mathbf{E}_P'$ belonging to the original target region $\mathbf{\Omega}$ finish the above processing in equation (22) as $\rho''(i, j)$, the final inpainted, encrypted image $\mathbf{E}_P''$ with patch filling and region smoothing can be obtained, which can be then transmitted to the content owner or authorized receiver for decryption, and the privacy-preserving inpainting for the outsourced image is completed.

Image decryption

After receiving the final inpainted, encrypted image $\mathbf{E}_P''$ from image restorer, the content owner or the authorized receiver conducts image decryption through the private key $\kappa_s = (p, q, \lambda)$ of Paillier encryption in equation (3)

$$\begin{aligned} m'(i,j) &= \frac{[\rho''(i,j)^{\lambda} \bmod n^2] - 1}{[g^{\lambda} \bmod n^2] - 1} \bmod n, \\ &\quad \text{for } \forall (i,j) \in \mathbf{E}_P'' \end{aligned} \tag{23}$$

where $m'(i, j)$ denotes the decrypted result of $\rho''(i, j)$ based on Paillier[36] cryptosystem. Then, since the weight vector $\mathbf{W}'_{i,j}$ is multiplied by $(4 + \omega)$ during the weighted mean filtering (refer to section "Region

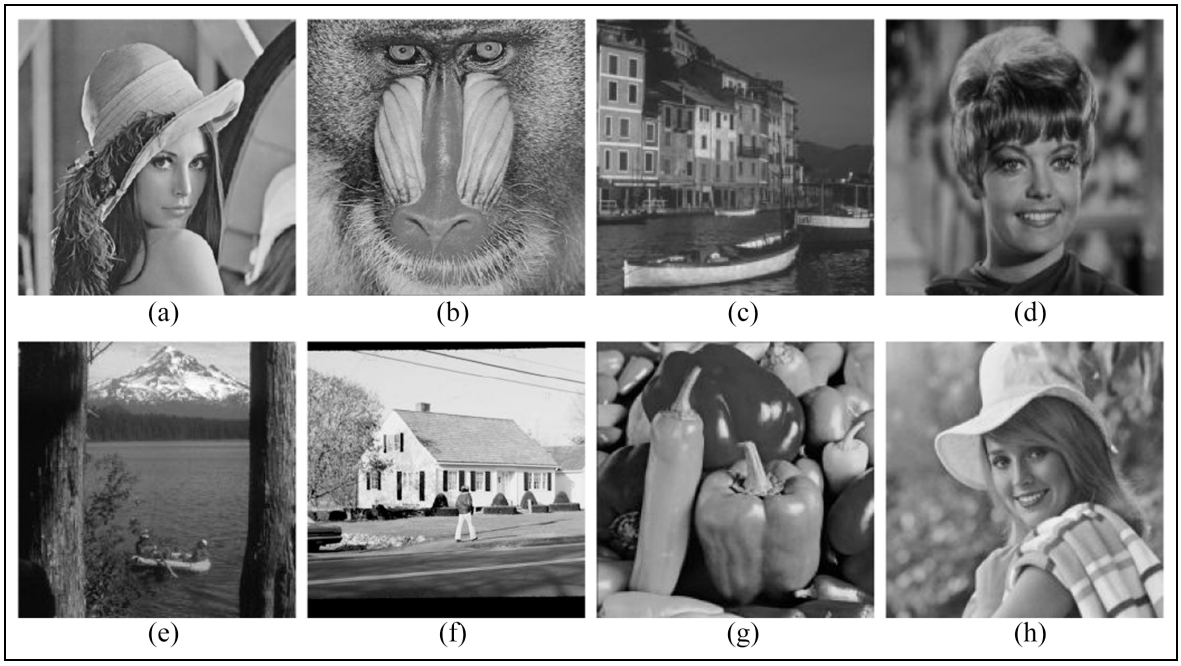

Figure 4. Some of the test images. (a) *Lena*. (b) *Baboon*. (c) *Portofino*. (d) *Zelda*. (e) *Lake*. (f) *House*. (g) *Peppers*. (h) *Elaine*.

smoothing with weighted mean filtering"), hence, all decrypted results $m'(i, j)$ should be further processed to produce the final inpainted image $\mathbf{I}_r$

$$m''(i,j) = \begin{cases} \dfrac{m'(i,j)}{4+\omega}, & \text{for } \forall (i,j) \in \Omega, \\ m'(i,j), & \text{for } \forall (i,j) \notin \Omega, \end{cases} \tag{24}$$

where $m''(i, j)$ denotes the value of the pixel at coordinate (i, j) of the final inpainted image $\mathbf{I}_r$ in the plaintext domain.

Experimental results and analysis

To demonstrate the effectiveness and superiority of the proposed scheme, a large number of test images were applied to conduct privacy-preserving image inpainting in the encrypted domain. In addition, we also compared the performance of inpainted quality between our scheme and some typical image inpainting schemes in the plaintext domain. Note that, in real applications, we often do not have original intact images. In the experiments, original intact images were just used to generate corresponding damaged images and to evaluate the visual quality of inpainted results as the references. Some of the images used in experiments and analysis, including *Lena, Baboon Portofino, Zelda, Lake, House, Peppers*, and *Elaine*, are illustrated in Figure 4.

Examples of the proposed scheme

In the proposed scheme, the inpainting operation is performed on the encrypted, damaged image $\mathbf{E}_P$ of Paillier homomorphic encryption. Figures 5 and 6 illustrate the encrypted results for the two standard test images *Lena* and *Baboon* both with the sizes of 512 $\times$ 512, in which subfigures (a) and (b) are the original images and their corresponding mask images $\mathbf{\Theta}$ for locating the damaged areas (target regions $\mathbf{\Omega}$ were marked as black), and subfigures (c) and (d) are the damaged images in the plaintext domain and the Paillier-encrypted domain, respectively. The damaged rates τ, which are defined as the ratios between the numbers of black pixels and all pixels in the mask images $\mathbf{\Theta}$, for *Lena* and *Baboon* are 4.93% and 5.20%, respectively. Note that, pixel values of the encrypted images in Figures 5 and 6(d) are the results after modulo 256 for displaying. It can be clearly observed from Figures 5 and 6 that, the contents of original images are concealed after encryption, which can protect the privacy of the content owner effectively.

After image encryption, the encrypted, damaged images were outsourced to image restorer for inpainting in the encrypted domain through the method described in section "Inpainting for encrypted image" (including patch filling and region smoothing). Then, the inpainted results in encrypted domain were sent back to the content owner or authorized receiver. Finally, after image decryption, the inpainted images in the plaintext domain were acquired. Figures 7 and 8(a) show the inpainted images in the encrypted domain for the two encrypted, damaged images *Lena* and *Baboon* in Figures 5 and 6(d), respectively, and their corresponding inpainted results in the plaintext domain (i.e. after image decryption) are presented in Figures 7 and 8(b). In our experiments, the two typical indices, peak

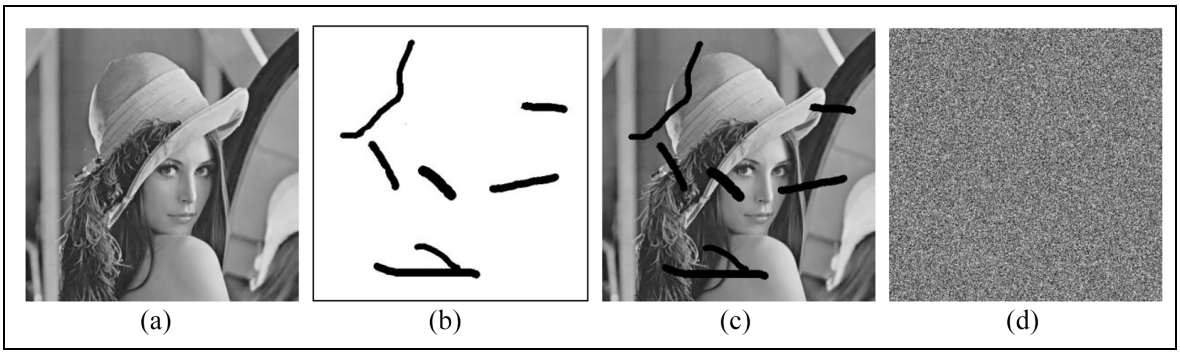

Figure 5. Encrypted results based on Paillier homomorphic encryption for *Lena*. (a) Original image. (b) Mask image Θ. (c) Damaged image $\mathbf{I}_d$ (τ = 4.93%). (d) Encrypted, damaged image $\mathbf{E}_P$

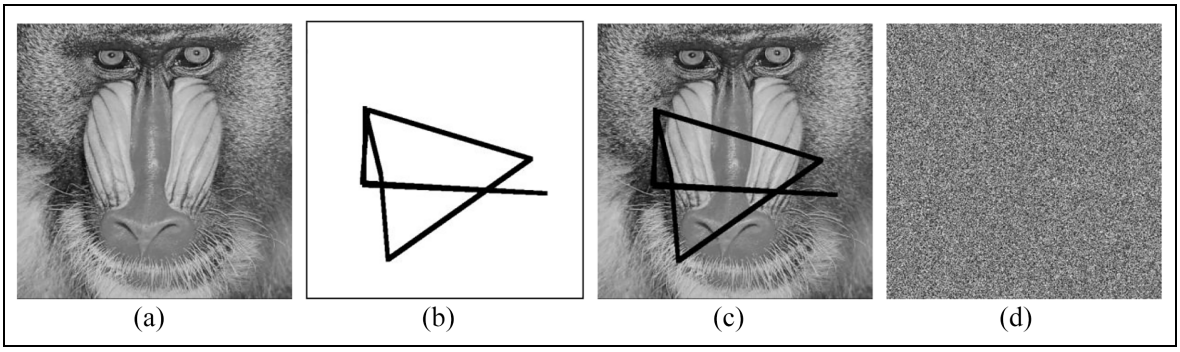

Figure 6. Encrypted results based on Paillier homomorphic encryption for *Baboon*. (a) Original image. (b) Mask image Θ. (c) Damaged image $\mathbf{I}_d$ (τ = 5.20%). (d) Encrypted, damaged image $\mathbf{E}_P$

signal-to-noise ratio (PSNR) and structural similarity (SSIM),[42] were utilized to evaluate visual quality of inpainted images in plaintext domain. Obviously, the greater the values of PSNR and SSIM are, the better visual quality of the inpainted image $\mathbf{I}_r$ with respect to the original intact image $\mathbf{I}_o$ is. PSNR values of the inpainted images *Lena* and *Baboon* in Figures 7 and 8(b) were calculated as 44.65 and 42.56 dB, and their SSIM values were 0.9826 and 0.9615, respectively. It can be found from Figures 7 and 8(b) and the values of PSNR and SSIM that, the visual quality of inpainted results for our scheme is satisfactory.

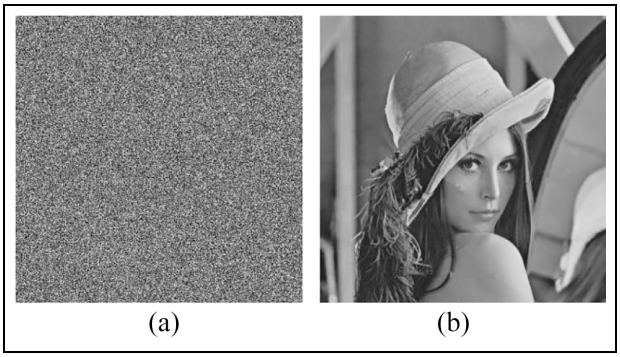

Figure 7. Inpainted results for *Lena*. (a) Inpainted image $\mathbf{E}_P''$ in encrypted domain. (b) Inpainted image $\mathbf{I}_r$ after decryption (PSNR = 44.65 dB, SSIM = 0.9826).

Parameter analysis

In our encrypted-image inpainting scheme, there are four main parameters: s and k in JL transform (section "Encryption with JL transform"), the size $l \times l$ of the block for priority calculation and matching (section "Block priority calculation"), and the size $r \times r$ of source region for conducting block matching (section "Block matching and patch filling"). The influence of inpainting performance with respect to these parameters is analyzed detailedly in the following. The experiments for parameter analysis in this subsection were conducted on the standard images sized 512 × 512. Besides *Lena* (τ = 4.93%) and *Baboon* (τ = 5.20%) in Figures 5 and 6(c), the other two standard images *Portofino* and *Zelda* were also used and their damaged rates τ are 3.98% and 3.95% with random forms, respectively.

Table 1. Visual quality of inpainted results under different s and k of JL transform ($l = 11$, $r = 33$).

Parameters		*Lena*		*Baboon*		*Portofino*		*Zelda*	
s	k	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
25	9	43.36	0.9775	40.70	0.9495	43.42	0.9784	42.79	0.9755
25	7	43.51	0.9778	40.51	0.9522	43.43	0.9780	42.95	0.9746
25	5	43.39	0.9781	40.48	0.9483	43.34	0.9779	42.92	0.9750
25	3	43.39	0.9789	40.47	0.9469	43.40	0.9783	42.49	0.9750
25	1	43.31	0.9799	40.55	0.9511	43.34	0.9773	42.89	0.9744
9	9	44.09	0.9819	41.06	0.9527	45.40	0.9825	44.22	0.9764
9	7	44.12	0.9816	41.32	0.9479	45.74	0.9830	44.23	0.9764
9	5	44.10	0.9813	41.06	0.9509	45.53	0.9826	44.04	0.9736
9	3	44.26	0.9814	41.08	0.9492	45.73	0.9833	43.98	0.9745
9	1	44.23	0.9814	41.27	0.9553	45.65	0.9823	44.17	0.9751
5	9	44.51	0.9831	41.65	0.9616	45.49	0.9841	44.54	0.9778
5	7	44.65	0.9826	41.64	0.9589	45.45	0.9840	44.40	0.9803
5	5	44.57	0.9831	41.70	0.9615	45.70	0.9837	44.37	0.9769
5	3	44.46	0.9831	41.62	0.9665	45.74	0.9841	44.09	0.9788
5	1	44.55	0.9828	41.42	0.9641	45.78	0.9837	44.04	0.9774

PSNR: peak signal-to-noise ratio; SSIM: structural similarity.

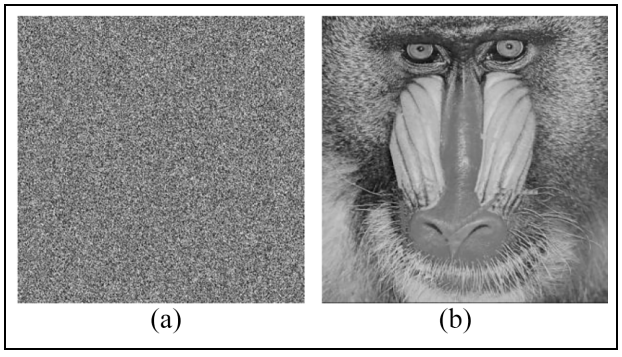

Figure 8. Inpainted results for *Baboon*. (a) Inpainted image $\mathbf{E}_P''$ in encrypted domain. (b) Inpainted image $\mathbf{I}_r$ after decryption (PSNR = 42.56 dB, SSIM = 0.9615).

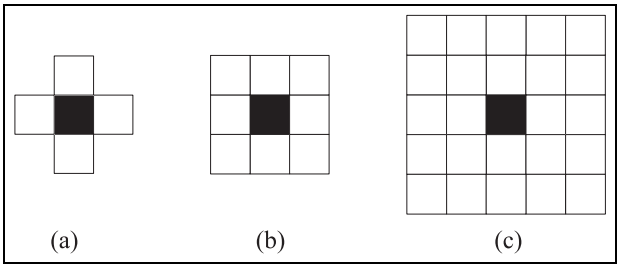

Figure 9. Three local patterns used in image encryption based on JL transform. (a) $s = 5$. (b) $s = 9$. (c) $s = 25$.

Parameters s and k in JL transform. During the image encryption of JL transform, a random matrix $\Re$ sized $k \times s$ is utilized, see equation (6), which can securely transform a s-pixels local pattern into a k-dimension vector for each center pixel $m(i, j)$ and also retain the Euclidean distance. Based on a large number of experiments, we find that, the parameter s has obvious influence on the visual quality of inpainted results because the distance for block matching is significantly changed with different values of s, while the influence of parameter k is not obvious. In Table 1, with the values of s equaling to 5, 9, and 25, the indices of PSNR and SSIM for the inpainted images, including *Lena, Baboon, Portofino*, and *Zelda*, are given, and the three local patterns in Figure 9(a)–(c) correspond to $s = 5$, $s = 9$, and $s = 25$, respectively. It can be concluded from the results that, generally speaking, the smaller the value of s is, the better quality of inpainted results can be obtained. In the process of image encryption with JL transform, each ciphertext is determined by all the s-pixels in the local pattern, thus, smaller patterns can reflect local information of plaintext pixels more accurately. In addition, we can observe from the results that, for most images, the visual quality of inpainted results were satisfactory when s was set as 5. However, when s was set as a larger value, such as 9 or 25, the Euclidean distance between two JL-encrypted pixels covered more unnecessary information of adjacent pixels, which led to inaccurate results in searching the best-matching block, and degraded the inpainted quality.

Parameter of size $l \times l$ for block priority calculation and block matching. The procedure of encrypted-image inpainting described in sections "Block priority calculation" and "Block matching and patch filling" is performed in the blockwise manner. For each pixel c in the target region Ω, a block $\mathbf{B}_c$ sized $l \times l$ with c at the center is exploited for priority calculation and block matching. Therefore, the block size $l \times l$ is also a key factor that

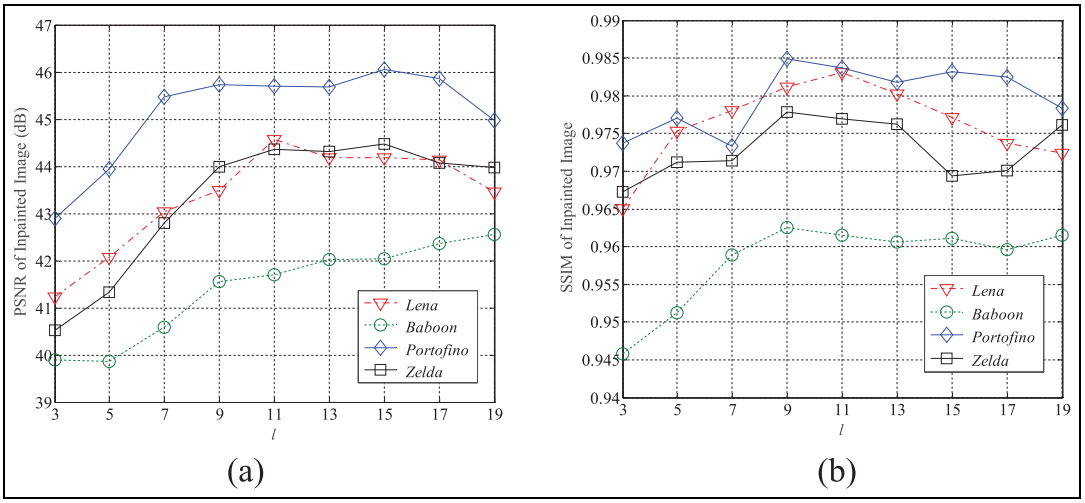

Figure 10. Visual quality of inpainted results under different sizes $l \times l$ of block $\mathbf{B}_c$ ($s = 5$, $k = 5$, $r = 33$). (a) PSNR of inpainted results. (b) SSIM of inpainted results.

has influence on the inpainting performance. Detailedly, too smaller size of block $\mathbf{B}_c$ may cause the loss of texture features in the inpainted result, while too larger size leads to the serious mosaic effect probably. It can be observed from Figure 10 that, for most images sized 512×512, the visual quality of inpainted results decreases rapidly when l is smaller than 11, the inpainted quality has a downward trend when l is greater than 17, and the inpainted quality tends to be relatively stable when l is in the range between 11 and 17.

Parameter of size $r \times r$ for source region $\mathbf{\Phi}$. In our scheme, the source region Φ for searching the best-matching block can be defined as the entire image excluding the target region $\mathbf{\Omega}$. However, for the trade-off of acceptable inpainted quality and feasible computation complexity, rather than the entire image, we only set a dilated window centered at the current pixel c and sized $r \times r$ as the source region Φ. The correlation between the block $\mathbf{B}_{c*}$ and all candidate blocks in source region $\mathbf{\Phi}$ for matching is inversely proportional to their distances; hence, too larger size of source region $\mathbf{\Phi}$ cannot guarantee the satisfactory repaired results and may make the inpainted region significantly different with its close neighborhood. However, too smaller size of source region $\mathbf{\Phi}$ may lead to the failure of finding the appropriate matching block and also make repaired results unsatisfactory. Therefore, a suitable size of source region Φ should be chosen for searching the best-matching block. However, from the results in Figure 11, we can find that the most appropriate sizes of source region Φ leading to the best inpainted quality fluctuate irregularly in a wide range for different images. Therefore, the size of source region Φ is better to be manually chosen through trial and error.

Comparisons with plaintext-domain inpainting schemes

To demonstrate the effectiveness of our scheme, we compared the decrypted, inpainted results of our proposed ciphertext-domain inpainting scheme with those of the three famous plaintext-domain inpainting schemes, that is, TV-based scheme by Shen and Chan,[17] Navier–Stokes PDE-based scheme by Bertalmio et al.,[2] and exemplar-based scheme by Criminisi et al.[20]

Figure 12 shows the inpainted results for *Lena, Baboon, Lake*, and *House*, in which the first column to the fifth column denote the damaged images, the results of the proposed scheme after image decryption, scheme by Shen and Chan,[17] scheme by Bertalmio et al.,[2] and scheme by Criminisi et al.,[20] respectively. The damaged images *Lena* in Figure 12(a1) and *Baboon* in Figure 12(a2) are the same as those in Figures 5(c) and 6(c) with damaged rates $\tau = 4.93\%$ and $\tau = 5.20\%$, respectively. Besides the conventional damage forms (such as lines and scratches in *Lena* and *Baboon*) for inpainting that can be considered as region filling, the other two typical examples of object removal for damaged images are presented in Figure 12(a3) and (a4). Detailedly, Figure 12(a3) shows the damaged image *Lake* imposed with a number of texts ($\tau = 4.86\%$). Figure 12(a4) shows the image *House* with a man as an undesirable foreground object, and to seamlessly remove the man from the image, this object was marked as the black region ($\tau = 0.96\%$). Note that, inpainting procedure of the schemes[2,17,20] was just performed on the plaintext-domain, damaged images in

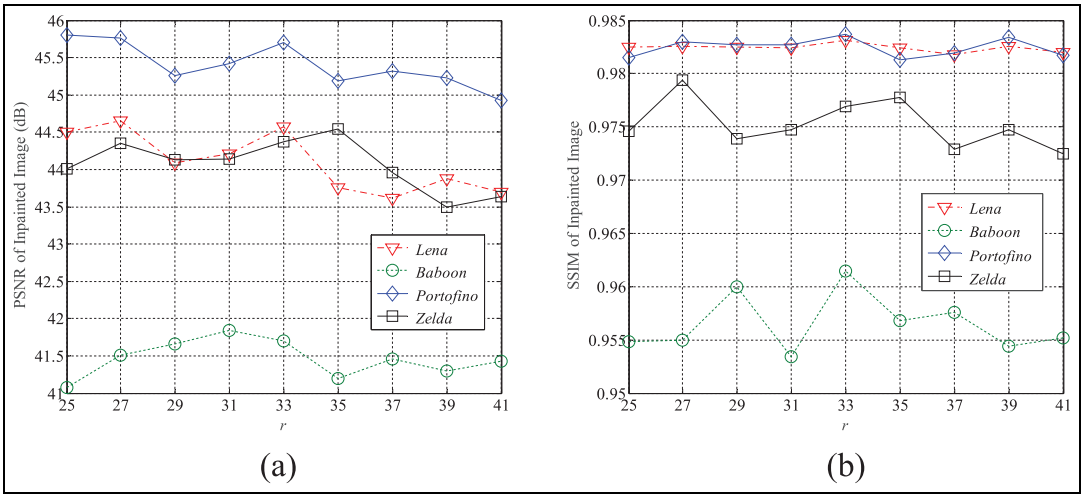

Figure 11. Visual quality of inpainted results under different sizes $r \times r$ for source region Φ ($s = 5$, $k = 5$, $l = 11$). (a) PSNR of inpainted results. (b) SSIM of inpainted results.

Figure 12. Comparison results for our encrypted-image inpainting and plaintext-image inpainting.[2,17,20] (a1)–(a4) Damaged images *Lena, Baboon, Lake* and *House*. (b1)–(b4) Results of our scheme after decryption. (c1)–(c4) Results of total variation-based scheme.[17] (d1)–(d4) Results of Navier–Stokes PDE-based scheme.[2] (e1)–(e4) Results of exemplar-based scheme.[20]

Table 2. Comparisons of {PSNR, SSIM} for the inpainted results between the proposed scheme and the schemes in Shen and Chan,[17] Bertalmio et al.,[2] and Criminisi et al.[20]

Images	Damaged rate (%)	Scheme in Shen and Chan[17]	Scheme in Bertalmio et al.[2]	Scheme in Criminisi et al.[20]	Proposed scheme
Lena	4.93	{34.25, 0.9831}	{43.17, 0.9811}	{44.99, 0.9805}	{44.65, 0.9826}
Baboon	5.20	{32.32, 0.9706}	{39.45, 0.9555}	{42.20, 0.9676}	{42.56, 0.9615}
Lake	4.86	{30.79, 0.9779}	{45.83, 0.9877}	{43.41, 0.9752}	{43.58, 0.9822}
Portofino	3.98	{37.98, 0.9873}	{42.10, 0.9771}	{45.57, 0.9780}	{46.06, 0.9832}
Zelda	3.95	{38.49, 0.9878}	{43.14, 0.9841}	{44.52, 0.9841}	{44.54, 0.9778}
Peppers	3.01	{39.77, 0.9943}	{43.04, 0.9850}	{47.52, 0.9884}	{46.74, 0.9864}
Elaine	4.19	{38.28, 0.9896}	{39.87, 0.9701}	{44.10, 0.9822}	{45.63, 0.9711}
Average	4.30	{35.98, 0.9844}	{42.37, 0.9772}	{44.61, 0.9794}	{44.82, 0.9778}

Figure 12(a1)–(a4), while the inpainting procedure of the proposed scheme was performed on their Paillier-encrypted, damaged versions.

We can observe from Figure 12 that the structural and textural information in the target region, such as lines and edges in *Lena* and fur and whiskers in *Baboon*, are successfully repaired by our scheme and there are not obvious blurring effects. Compared with the three plaintext-domain schemes,[2,17,20] our scheme can generally obtain better visual quality of inpainted results than the PDE-based schemes[2,17] and achieve the comparable performance with the texture synthesis-based scheme.[20] It can also be found that our scheme can seamlessly remove the imposed texts for *Lake* and effectively inpaint the occluded window, bushes, and lawn for *House* without a priori model. The values of PSNR and SSIM for the inpainted results for *Lena, Baboon*, and *Lake* under the four schemes are listed in the first three rows of Table 2 (PSNR and SSIM for *House* are not available due to the absence of original reference image).

In addition to the above inpainting results in Figure 12 for the four images (*Lena, Baboon, Lake*, and *House*), four other standard test images, including *Portofino, Zelda, Peppers*, and *Elaine*, were also involved in the experiments. The damaged forms for these four images were random, and the corresponding damaged rates τ were 3.98%, 3.95%, 3.01%, and 4.19%, respectively. Table 2 gives a summary of PSNR and SSIM for the seven inpainted images with respect to their corresponding original images under the three plaintext-domain inpainting schemes[2,17,20] and the proposed ciphertext-domain inpainting scheme. It can be found that our scheme is more suitable for repairing the images with richer textures and larger damaged area. In a word, although the inpainting procedure of our scheme is implemented in ciphertext domain, it can achieve comparable inpainting performance with plaintext-domain schemes and can also realize privacy preserving for the content owner effectively.

Applications in color images and comparison with PatchMatch

Besides gray-level images, we also applied our scheme in color images for privacy-preserving inpainting. Test images were randomly selected from the ImageNet database, which is a famous large-scale image database in the fields of image recognition and computer vision, and a representative inpainting scheme for color images in plaintext domain, called PatchMatch,[21] was utilized for comparison, see Figure 13. The first column of Figure 13 denotes the original images randomly chosen from ImageNet, and the second column is their damaged versions with various damaged rates. The third column and the fourth column are the inpainted results of our scheme (after image decryption) and PatchMatch,[21] respectively. Note that, PatchMatch[21] was just performed on the damaged images in plaintext domain, that is, Figure 13(b1)–(b4), while our scheme was performed on the Paillier-encrypted, damaged versions of Figure 13(b1)–(b4). It can be observed from Figure 13 that the structural and textural information in the target region, such as lines and textures, was basically repaired by our scheme, and there were not obvious blurring effects. Therefore, our scheme can not only achieve comparable visual quality of inpainted results with the plaintext-domain scheme for color images, that is, PatchMatch,[21] but also can realize privacy protection for image contents from content owner since our inpainting operation of image restorer is processed on encrypted versions of damaged images.

Discussions of security and computational complexity

Security discussions. Since Paillier[36] proved that Paillier encryption provides semantic security and the attacker cannot learn any information of the plaintext except for the length of the plaintext, hence, we mainly discuss the security of JL transform used in the scheme. In our scheme, the encrypted result of each pixel by JL

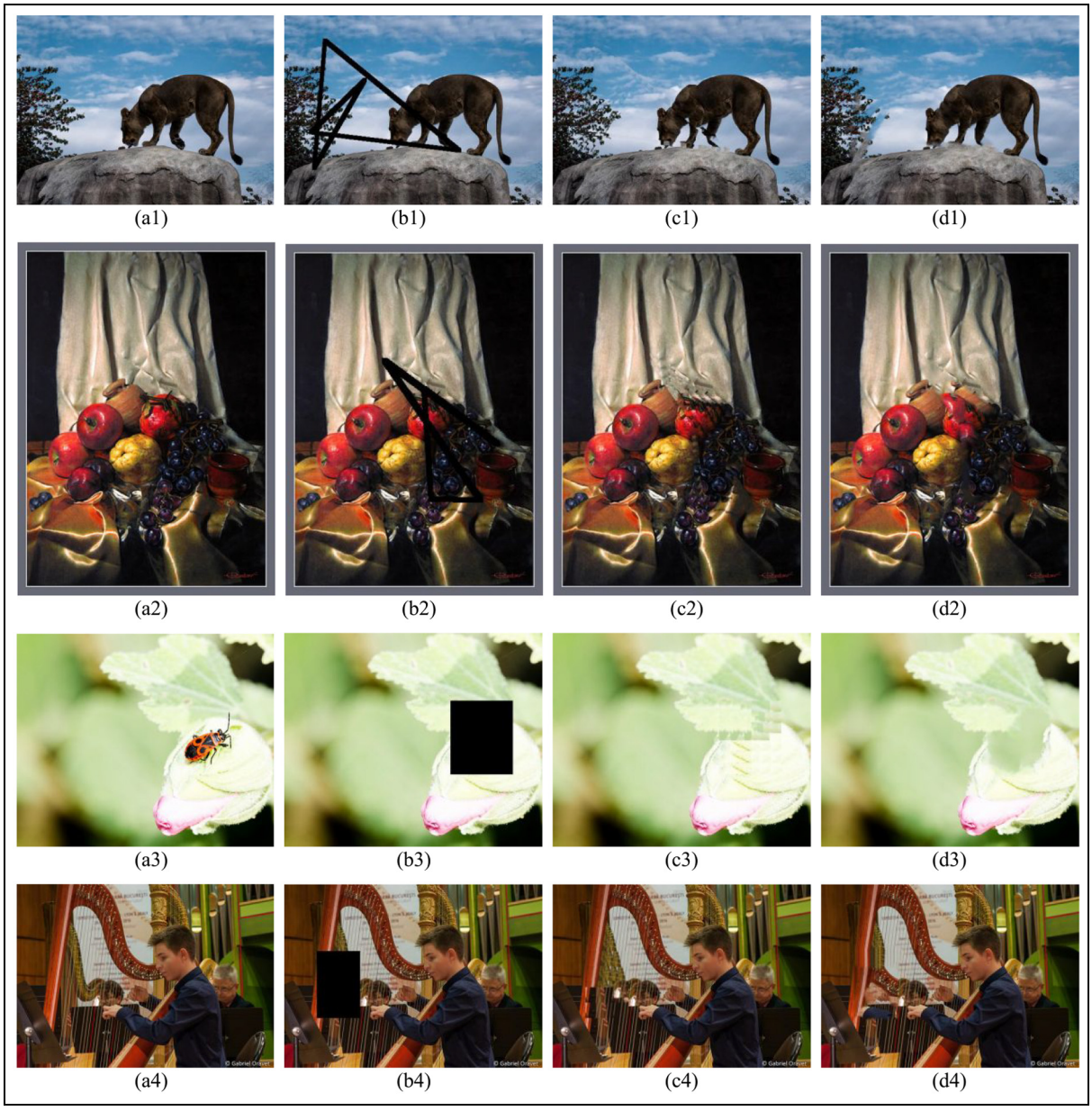

Figure 13. Inpainting application for color images and comparison between our encrypted-image inpainting with the plaintext-image PatchMatch inpainting.[21] (a1)–(a4) Original images from ImageNet. (b1)–(b4) Damaged versions of (a1)–(a4). (c1)–(c4) Results of our scheme after decryption. (d1)–(d4) Results of PatchMatch.[21]

transform is a vector instead of a single value. After all $H \times W$ pixels $m(i, j)$ in $\mathbf{I}_{\mathrm{d}}$ are encrypted through equation (6), the content owner collects all corresponding encrypted results $\mathbf{L}_{i,\ j}$ to form the encrypted image $\mathbf{E}_{\mathrm{J}}$. Binarization attack mentioned in Hu et al.[29] means that for each JL-encrypted result $\mathbf{L}_{i,\ j}$ with respect to the plaintext pixel $m(i,\ j)$, which is a k-dimension vector, calculate the mean value of the k components in $\mathbf{L}_{i,\ j}$, thus, corresponding to $H \times W$ plaintext pixels, $H \times W$ mean values can be obtained, which preserve the relative positions and intensities of pixels in the plaintext image $\mathbf{I}_{\mathrm{d}}$. Then, the attacker can easily quantize these $H \times W$ mean values into binary values through setting a threshold and reshape these $H \times W$ binary values to produce a binary image sized $H \times W$, which is visually approximate to $\mathbf{I}_{\mathrm{d}}$ and leaks the privacy of image owner.

To defeat the potential binarization attack, we should eliminate the correlation between neighboring pixels. Hence, after Paillier encryption and JL transform, random permutation should be further performed on the two encrypted images $\mathbf{E}_{\mathrm{P}}$, $\mathbf{E}_{\mathrm{J}}$, and their corresponding mask image $\boldsymbol{\Theta}$. It should be noted that our inpainting scheme is based on patch filling, and

Table 3. Execution time of different operations (seconds).

Image owner/sender	Image size	Paillier encryption	JL transform	Random permutation
	256×256	1.936	0.185	1.172
	512×512	7.752	0.743	4.604
Image owner/receiver	Image size	Inverse permutation		Paillier decryption
	256×256	0.483		6.325
	512×512	1.956		25.169

JL: Johnson–Lindenstrauss.

damaged region is repaired patch by patch rather than pixel by pixel. Hence, in our scheme, the random permutation is conducted on $\mathbf{E}_P$, $\mathbf{E}_J$, and $\mathbf{\Theta}$ in a blockwise manner. In addition, to keep the corresponding relationships for the encrypted results of $\mathbf{E}_P$, $\mathbf{E}_J$, and $\mathbf{\Theta}$, the same pseudo-random permutation sequence derived by secret key κ_r should be utilized for them. Thus, without knowing the secret key κ_r, the attacker may try all possible permutations and the number of all possible permutations is $(H \times W / l^2)!$ For an image sized 512×512 ($H = W = 512$) and the block size equaling 9×9 ($l = 9$), the possibility for the attacker to crack the correct permutation is lower than 1/3236! Therefore, without the secret key κ_r, even if the attacker implements the binarization attack on the ciphertext encrypted by JL transform, it is difficult to recover approximate content of original image $\mathbf{I}_d$ because the locations of image blocks are completely permuted.

However, since local correlation in the permuted image is removed after random permutation, the source region $\mathbf{\Phi}$ sized $r \times r$ for candidate block searching will no longer work. An effective way to solve this problem is directly extending the source region $\mathbf{\Phi}$ sized $r \times r$ to the whole image sized $H \times W$. In other words, block matching will be conducted within the whole image instead of the source region, which may increase computational complexity significantly. Hence, we can adopt the random sampling strategy in Chan et al.[43] to reduce computational complexity, which only randomly calculates a small number of image block distances according to sampling patterns.

Computational complexity. In our scheme, both JL Transform and Paillier encryption cause the ciphertext expansion. To reduce the storage on the client side, the image pixels can be encrypted one by one, and each encrypted pixel is directly transmitted to the cloud immediately. Thus, the client side does not need to store the whole encrypted images. As for Paillier encryption indicated in equation (4), the total data volume transmitted from the client side to the cloud is about $2|n| \times H \times W$ bits, where $|n|$ denotes the bit length of n, and $H \times W$ is the number of image pixels. Thus, for a 1024-bit integer n, the ciphertext expansion ratio of Paillier encryption in our scheme is about $2 \times 1024 / 8 = 256$ for one 8-bit pixel. Since the encrypted result of each pixel $m(i, j)$ for JL transform is a k-dimension vector instead of a single value, hence, the ciphertext expansion ratio of JL transform is approximately equal to the parameter k in our scheme. After the cloud completes the operation of image inpainting in encrypted domain, the final inpainted, Paillier-encrypted image $\mathbf{E}_P''$ should be transmitted from the cloud to the client for decryption, while the ciphertext $\mathbf{E}_J$ encrypted by JL transform does not need to be sent back to the client.

As for execution time, the most complicated computation on the client side is Paillier encryption and decryption. To increase the efficiency, we adopted the fast algorithm proposed by Jost et al.,[44] which is a variant of Paillier cryptosystem based on pre-computing and look-up table. The execution time for the different operations on the sender side and the receiver side is listed in Table 3, which includes Paillier encryption, JL transform, random permutation for image owner/sender, and Paillier decryption and inverse permutation for legal receiver. The pre-computing before encryption consumes 9.56 s. Note that, JL transform is only required on the sender side, and random permutation is conducted for $\mathbf{E}_P$, $\mathbf{E}_J$, and $\mathbf{\Theta}$ on the sender side and only for $\mathbf{E}_P''$ on the receiver side. All experiments were implemented on a personal computer with a 3.20 GHz Intel i5 processor, 4 GB memory, and Windows 7 operating system.

Conclusion

In this work, we propose a new privacy-preserving inpainting framework for outsource image and a specific encrypted-image inpainting scheme, which can not only repair scratches and remove undesirable objects in images seamlessly but also can protect the user privacy through encrypting image contents toward image restorer. Paillier encryption and JL transform are both applied by content owner to generate encrypted results for the damaged image. The inpainting mechanism of our scheme is based on block matching and patch

filling for the marked target region according to priority order. Since JL transform can preserve Euclidean distance between two vectors before and after encryption, the best-matching block with the smallest distance to current block can be found in source region and utilized for patch filling in the Paillier-encrypted image by image restorer. To further eliminate the possible mosaic effect after decryption, weighted mean filtering for the intermediate, inpainted image in encrypted domain is conducted based on homomorphic properties of addition in Paillier cryptosystem. After image decryption with correct secret key, content owner or authorized receiver can obtain the final inpainted image in plaintext domain. Experimental results demonstrate that the proposed encrypted-image inpainting scheme not only achieves comparable performance of inpainted-image quality with those typical inpainting schemes for plaintext images but also realizes privacy preserving for image content owner effectively due to the complete encrypted-domain operation.

Our current work belongs to non-learning-based inpainting method with privacy-preserving capability. However, learning-based methods can achieve superior global semantic structures of inpainted results, especially with rapid development of deep learning technique. Therefore, in the future, how to effectively and efficiently achieve the learning-based, privacy-preserving image inpainting deserves in-depth investigations. Some works studied privacy-preserving deep learning with homomorphic cryptosystem[45,46] and utilized asynchronous stochastic gradient descent as applied to neural networks, combining with the additively homomorphic encryption, which are helpful to our future work. In addition, functional encryption supports the restricted secret keys that enable a key holder to learn a specific function of encrypted data, but learn nothing else about the data,[47] which may be suitable well to solve the problem of block matching (i.e. a specific function) in encrypted domain. Hence, further works also include realizing encrypted block matching through functional encryption and achieving the noise-resisting robustness of privacy-preserving inpainting.

Acknowledgement

The authors thank the anonymous reviewers for their valuable suggestions.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by the National Natural Science Foundation of China (Grant No. 61902239) and in part by the Research Fund of Guangxi Key Lab of Multi-Source Information Mining and Security (MIMS20-03).

ORCID iD

Ching-Chun Chang https://orcid.org/0000-0002-7539-1451

References

1. Bertalmio M, Sapiro G, Caselles V, et al. Image inpainting. In: *Proceedings of the 27th annual conference on computer graphics and interactive techniques (SIGGRAPH)*, New Orleans, LA, 23–28 July 2000, pp.417–424. New York: ACM.
2. Bertalmio M, Bertozzi AL and Sapiro G. Navier-stokes, fluid dynamics, and image and video inpainting. In: *Proceedings of the IEEE international conference on computer vision and pattern recognition*, Kauai, HI, 8–14 December 2001, pp.355–362. New York: IEEE.
3. Criminisi A, Perez P and Toyama K. Object removal by exemplar-based inpainting. In: *Proceedings of the IEEE international conference on computer vision and pattern recognition*, Madison, WI, 18–20 June 2003, pp.721–728. New York: IEEE.
4. Qin C, He ZH, Yao H, et al. Visible watermark removal scheme based on reversible data hiding and image inpainting. *Signal Process* 2018; 60: 160–172.
5. Ballester C, Bertalmío M, Caselles V, et al. An inpainting- based deinterlacing method. *IEEE T Image Process* 2007; 16(10): 2476–2491.
6. Liu D, Sun X, Wu F, et al. Image compression with edge-based inpainting. *IEEE T Circ Syst Vid* 2007; 17(10): 1273–1287.
7. Qin C, Chang CC and Chiu YP. A novel joint data-hiding and compression scheme based on SMVQ and image inpainting. *IEEE T Image Process* 2014; 23(3): 969–978.
8. Wang SS and Tsai SL. Automatic image authentication and recovery using fractal code embedding and image inpainting. *Pattern Recogn* 2008; 41(2): 701–712.
9. Qin C, Chang CC and Chen KN. Adaptive self-recovery for tampered images based on VQ indexing and inpainting. *Signal Process* 2013; 93(4): 933–946.
10. Qin C, Chang CC, Huang YH, et al. An inpainting-assisted reversible steganographic scheme using a histogram shifting mechanism. *IEEE T Circ Syst Vid* 2013; 23(7): 1109–1118.
11. Rane SD, Sapiro G and Bertalmio M. Structure and texture filling-in of missing image blocks in wireless transmission and compression applications. *IEEE T Image Process* 2003; 12(3): 296–303.
12. Shih TK, Lu LC, Wang YH, et al. Multi-resolution image inpainting. In: *Proceedings of the IEEE international conference on multimedia and expo*, Baltimore, MD, 6–9 July 2003, vol. 1, pp.485–488. New York: IEEE.
13. Shih TK, Chang RC and Lu LC, et al. Adaptive digital image inpainting. In: *Proceedings of the 18th international conference on advanced information networking and*

applications, Fukuoka, Japan, 29–31 March 2004, vol. 1, pp.71–76. New York: IEEE.
14. Telea A. An image inpainting technique based on the fast marching method. *J Graph Tool* 2004; 9(1): 23–34.
15. Qin C, Cao F and Zhang XP. Efficient image inpainting using adaptive edge-preserving propagation. *Imaging Sci J* 2011; 59(4): 211–218.
16. Chan TF and Shen JH. Non-texture inpainting by curvature-driven diffusions. *J Vis Commun Image R* 2001; 12(10): 436–449.
17. Shen JH and Chan TF. Mathematical models for local nontexture inpainting. *SIAM J Appl Math* 2002; 62(3): 1019–1043.
18. Qin C, Wang SZ and Zhang XP. Simultaneous inpainting for image structure and texture using anisotropic heat transfer model. *Multimed Tools Appl* 2012; 56(3): 469–483.
19. Drori I, Cohen-Or D and Yeshurun H. Fragment-based image completion. *ACM T Graphic* 2003; 22: 303–312.
20. Criminisi A, Perez P and Toyama K. Region filling and object removal by exemplar-based image inpainting. *IEEE T Image Process* 2004; 13(9): 1200–1212.
21. Barnes C, Shechtman E, Finkelstein A, et al. PatchMatch: a randomized correspondence algorithm for structural image editing. *ACM T Graphic* 2009; 28(3): 24.
22. Iizuka S, Simo-Serra E and Ishikawa H. Globally and locally consistent image completion. *ACM T Graphic* 2017; 36(4): 107.
23. Yu JH, Lin Z, Yang JM, et al. Generative image inpainting with contextual attention. In: *Proceedings of the 2018 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, Salt Lake City, UT, 18–23 June 2018, pp.5505–5514. New York: IEEE.
24. Liu GL, Reda FA, Shih KJ, et al. Image inpainting for irregular holes using partial convolutions. In: *Proceedings of the European conference on computer vision (ECCV)*, Munich, 8–14 September 2018, pp.89–105. Cham: Springer.
25. Pathak D, Krähenbühl P, Donahue J, et al. Context encoders: feature learning by inpainting. In: *Proceedings of the 2016 IEEE conference on computer vision and pattern recognition (CVPR)*, Las Vegas, NV, 26 June–1 July 2016, pp.2536–2544. New York: IEEE.
26. Ren K, Wang C and Wang Q. Security challenges for the public cloud. *IEEE Internet Comput* 2012; 16(1): 69–73.
27. Bianchi T, Piva A and Barni M. On the implementation of the discrete Fourier transform in the encrypted domain. *IEEE T Inf Foren Sec* 2009; 4(1): 86–97.
28. Qin C, Zhou Q, Cao F, et al. Flexible lossy compression for selective encrypted image with image inpainting. *IEEE T Circ Syst Vid* 2019; 29(11): 3341–3355.
29. Hu XJ, Zhang WM, Li K, et al. Secure nonlocal denoising in outsourced images. *ACM T Multim Comput* 2016; 12(3): 40.
30. Ren YL, Zhang XP, Feng GR, et al. How to extract image features based on co-occurrence matrix securely and efficiently in cloud computing. *IEEE T Cloud Comput* 2020; 8(1): 207–219.
31. Qin C, Zhang W, Cao F, et al. Separable reversible data hiding in encrypted images via adaptive embedding strategy with block selection. *Signal Process* 2018; 153: 109–122.
32. Qin C, Jiang CY, Mo Q, et al. Reversible data hiding in encrypted image via secret sharing based on GF(p) and GF(28). *IEEE T Circ Syst Vid*. Epub ahead of print June 2021. DOI: 10.1109/TCSVT.2021.3091319.
33. Efros AA and Leung TK. Texture synthesis by non-parametric sampling. In: *Proceedings of the IEEE international conference on computer vision*, Corfu, 20–27 September 1999, vol. 2, pp.1033–1038. New York: IEEE.
34. Rivest RL, Adleman L and Dertouzos ML. On data banks and privacy homomorphisms. In: DeMillo *RA (ed.) Foundations of secure computation*, vol. 32, issue no. 4. New York: Academic Press, 1978, pp.169–179.
35. ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE T Inform Theory* 1985; 31(4): 469–472.
36. Paillier P. Public-key cryptosystems based on composite degree residuosity classes. In: Stern J (ed.) *Advances in cryptology—EUROCRYPT'99*, vol. 1592 (Lecture notes in computer science). Berlin; Heidelberg: Springer, 1999, pp.223–238.
37. Damgård I and Jurik M. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: *Proceedings of the international workshop on public key cryptography*, Jeju Island, South Korea, 13–15 February 2001, vol. 1992, pp.119–136. Berlin; Heidelberg: Springer.
38. Gentry C. *A fully homomorphic encryption scheme*. PhD Thesis, Stanford University, Stanford, CA, September 2009.
39. Cheon JH and Kim J. A hybrid scheme of public-key encryption and somewhat homomorphic encryption. *IEEE T Inf Foren Sec* 2015; 10(5): 1052–1063.
40. Johnson WB and Lindenstrauss J. Extensions of Lipschitz mappings into a Hilbert space. *Contemp Math* 1984; 26: 189–206.
41. Kenthapadi K, Korolova A, Mironov I, et al. Privacy via the Johnson-Lindenstrauss transform. *J Privacy Confident* 2013; 5(1): 39–71.
42. Wang Z, Bovik AC, Sheikh HR, et al. Image quality assessment: from error visibility to structural similarity. *IEEE T Image Process* 2004; 13(4): 600–612.
43. Chan SH, Zickler T and Lu YM. Monte Carlo non-local means: random sampling for large-scale image filtering. *IEEE T Image Process* 2014; 23(8): 3711–3725.
44. Jost C, Lam H, Maximov A, et al. Encryption performance improvements of the Paillier cryptosystem, 2015, https://eprint.iacr.org/2015/864.pdf
45. Phong LT, Aono Y, Hayashi T, et al. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE T Inf Foren Sec* 2018; 13(5): 1333–1345.
46. Shokri R and Shmatikov V. Privacy-preserving deep learning. In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, Denver, CO, 12–16 October 2015, pp.1310–1321. New York: ACM.
47. Boneh D, Sahai A and Waters B. Functional encryption: a new vision for public-key cryptography. *Commun ACM* 2012; 55(11): 56–64.