Robust Privacy-Preserving Image Sharing over Online Social Networks (OSNs)

WEIWEI SUN and JIANTAO ZHOU, University of Macau
SHUYUAN ZHU, University of Electronic Science and Technology of China
YUAN YAN TANG, University of Macau

Sharing images online has become extremely easy and popular due to the ever-increasing adoption of mobile devices and online social networks (OSNs). The privacy issues arising from image sharing over OSNs have received significant attention in recent years. In this article, we consider the problem of designing a secure, robust, high-fidelity, storage-efficient image-sharing scheme over Facebook, a representative OSN that is widely accessed. To accomplish this goal, we first conduct an in-depth investigation on the manipulations that Facebook performs to the uploaded images. Assisted by such knowledge, we propose a DCT-domain image encryption/decryption framework that is robust against these lossy operations. As verified theoretically and experimentally, superior performance in terms of data privacy, quality of the reconstructed images, and storage cost can be achieved.

CCS Concepts: • **Security and privacy** → **Social network security and privacy**; • **Information systems** → **Multimedia information systems**;

Additional Key Words and Phrases: Online social networks, image sharing, privacy-preserving, JPEG

ACM Reference format:
Weiwei Sun, Jiantao Zhou, Shuyuan Zhu, and Yuan Yan Tang. 2018. Robust Privacy-Preserving Image Sharing over Online Social Networks (OSNs). *ACM Trans. Multimedia Comput. Commun. Appl.* 14, 1, Article 14 (January 2018), 22 pages.
https://doi.org/10.1145/3165265

1 INTRODUCTION

With the wide adoption of mobile devices equipped with high-resolution on-board cameras, online image sharing has become an extremely easy and popular activity. Currently, users can conveniently share images through online social networks (OSNs) such as Facebook and Google+ or dedicated image-sharing platforms such as Instagram. Take Facebook, for example: there are 1.23 billion *daily* active users, with 300 million images uploaded *daily* for December 2016 [10, 41]. Usually, these images contain a significant amount of personal information and, hence, are privacy

This work was supported in part by the Macau Science and Technology Development Fund under Grant No. FDCT/046/2014/A1 and Grant No. FDCT/022/2017/A1, in part by the Research Committee at the University of Macau under Grant No. MYRG2015-00056-FST and Grant No. MYRG2016-00137-FST, and in part by the National Science Foundation of China under Grant No. 61402547 and Grant No. 61672134.

Authors' addresses: W. Sun, J. Zhou (Corresponding author), and Y. Yan Tang are with the Department of Computer and Information Science, Faculty of Science and Technology, University of Macau; emails: {Yb57457, jtzhou, yytang}@umac.mo; S. Zhu is with the School of Electronic Engineering, University of Electronic Science and Technology of China; email: eezsy@uestc.edu.cn.

https://doi.org/10.1145/3165265

sensitive in nature. It seems that the OSN providers have not paid adequate attention to privacy protection issues, arousing much criticism from the public [16, 18, 20, 25, 42]. These shared images are not well protected in the sense that most of them can be freely browsed, downloaded, distributed, or even inappropriately used by third-party applications [7, 28, 29].

To offer privacy protection for images shared over OSNs, many works focused on designing access control protocols such that images can be accessed by only a select group of users. In the access control rules designed in [5, 17, 22, 27], the policies were applied at the image level; namely, an *entire* image was accessible or nonaccessible for a specific user. To provide fine-grained access control, [1, 6, 8, 15, 36] proposed to divide an image into several parts, each of which could have different access policies. For instance, the faces in an image could convey a more significant amount of sensitive information and should be applied with a higher level of protection. In contrast, a background scene is relatively less sensitive; thus, a low level of access control may be sufficient. As pointed out in [32], most of the existing privacy control strategies were far from adequate, leading to severe information leakage when users failed to understand the complex privacy settings or when OSNs cannot implement the access control policies correctly.

Another category of privacy protection schemes aimed at scrambling/encrypting the image data before uploading images to OSNs for sharing purposes. Essentially, stronger protection could be offered, as in this case even OSNs themselves cannot get the information of the original images. Along this line, a series of JPEG scrambling methods was designed [37–39]. Specifically, user-defined sensitive regions of an image were scrambled with different secret keys, which were then shared among different friends [37]. Here, the scrambling was achieved by randomly flipping the signs of the quantized DCT coefficients. Considering the undesirable confidentiality offered by the sign scrambling, this work was then extended to an ultra-high level of scrambling [38], which was achieved by performing bitwise XOR operations between DC coefficients and some generated keystreams. Furthermore, the traditional public key cryptography was employed in [39] to encrypt selected regions. Besides the above constructive approaches, Yuan et al. also pointed out the disadvantages of the existing general-purpose image-scrambling methods, e.g., [14, 30, 44], in terms of their efficiency, complexity, and format compatibility. In [12], a method called PUPPIES was developed, in which sensitive regions were encrypted by adding a random number on each DCT coefficient and then applying normalization. It should be noted that in the selective scrambling solutions, adversaries can still get a significant amount of information from the unprotected parts. In fact, a recent study showed that users were effective in recognizing their friends even in images in which their faces were not clearly visible [23].

Therefore, some other works focused on designing encryption schemes over the entire shared image, rather than over selected regions[1]. Ra et al. proposed the P3 [24], a privacy-preserving image-sharing system, which splits an image into a public part and a private part. The public part was shared over OSNs in the plaintext; the secret part was encrypted and stored in a cloud storage server. The introduction of an additional cloud server dramatically complicated the file management system and, in many situations, the cloud server itself cannot be fully trusted. More important, as will be clear soon, the public part of P3 would leak significant visual information of the original image. Alternatively, Tierney et al. designed a system called Cryptagram, enabling users to encrypt images with traditional block ciphers and then embed the encrypted bitstream into a JPEG file. However, the desirable properties of Cryptagram were realized at the cost of a dramatically increased storage burden on OSNs, as the use of cover images resulted in significant file expansion. In addition, Dirik and Memon devised a robust image encryption for social

[1]These full-frame image encryption methods could also be readily applied to encrypt selected regions once they were specified.

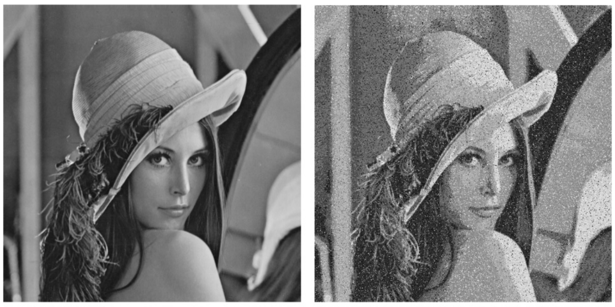

(a) Original Lena (b) Reconstructed Lena

Fig. 1. The reconstructed result of the online shared image with AES encryption.

networks, where the encryption was achieved by shuffling the middle and high DCT frequencies [9]. To handle the unexpected dark and light tones in the encrypted images, they also suggested limiting the power of DCT coefficients by scaling with an empirically determined factor. Other relevant approaches were reported in [2, 13, 33, 40, 43].

In the above scrambling/encryption-based solutions for protecting the privacy of online shared images, a challenge that has been largely overlooked is the negative effect brought by the lossy operations conducted by OSNs. It is a well-known fact that OSNs apply various operations to the images uploaded by the users, e.g., to reduce the file size or to impose some format constraints. For instance, Facebook converts all uploaded images to JPEG files, choosing quality factor (QF) adaptively without user input. In other words, the links provided by OSNs cannot be regarded as perfect channels but rather as lossy ones. These lossy operations on the uploaded images actually significantly affect the protection strategies. For example, if we naïvely apply AES encryption on the images prior to sharing them, we observe severe distortions on the downloaded images. As shown in Figure 1, the PSNR value of the downloaded Lena is only 18.36 dB, which is clearly unacceptable.

It is therefore crucial to develop an end-to-end image encryption/decryption mechanism that is robust against these lossy operations. This is not a trivial task because of the following challenges: (1) the lossy operations conducted by OSNs are usually unknown and (2) users cannot control how OSNs apply these lossy operations. In fact, developing robust protection schemes has been less explored in the multimedia security community in the sense that most of the existing schemes assumed that the protected images simply go through a perfect channel to arrive at the users' side [35, 45].

In this work, we address the problem of designing a robust privacy-preserving image-sharing scheme over Facebook, a representative OSN that is widely accessed. For ease of presentation and without losing generality, we consider only 8-bit grayscale images. The proposed technique could be readily extended to the other OSNs and color images as well. To deal with the above design challenges, we first conduct an in-depth investigation on the manipulations that Facebook performs to the uploaded images. We find that four types of operations will be applied to the uploaded images: format conversion, resizing, JPEG compression, and enhancement filtering. The parameters employed in these operations are important and could be estimated through an offline training procedure. Being aware of the knowledge on the manipulations, we design a DCT-domain image encryption scheme that is robust against these lossy operations. To the best of our knowledge, it is the first design of a privacy-preserving image-sharing scheme that explicitly exploits the knowledge on the lossy operations conducted by OSNs. As expected and will be verified by our experimental results, superior performance in terms of data privacy, quality of the reconstructed images, and server side storage cost can be achieved.

***Difference from conference version*:** Portions of the work presented in this article have previously appeared in [31]. We have significantly revised and clarified the content of the article, and improved many technical details compared with [31]. The primary improvements can be summarized as follows. First, to solve the critical pixel overflow problem, we develop an optimization framework for optimally determining the shrinkage factors, leading to much better image reconstruction performance. Second, we augment the image encryption scheme by adding a pixel-domain block permutation module. This could significantly improve the visual security of the resulting encrypted image. Third, we include thorough security analysis to show that the proposed scheme is secure against various attacks. Finally, all the experiments in our performance evaluation in terms of security, reconstruction quality, and storage overhead are completely redone.

The rest of this article is organized as follows. Section 2 reviews the JPEG standard. In Section 3, we present our findings on the lossy operations conducted by Facebook on the shared images. Section 4 is devoted to the descriptions of our system model and design goals. We describe the core parts of our privacy-preserving image-sharing scheme in Section 5 and give the security analysis in Section 6. Experimental results are provided in Section 7 to show the superior performance of our proposed scheme. We present our conclusions in Section 8.

2 PRELIMINARY OF JPEG

In this section, we briefly review the JPEG standard, as it is highly related to our proposed privacy-preserving image-sharing scheme. For a detailed description, refer to [34].

Prior to JPEG compression, the input image is first partitioned into nonoverlapping 8×8 blocks. Each pixel block $\mathbf{x} = \{x_{r,s}\}$ is then subject to discrete cosine transform (DCT), producing the DCT coefficient block $\mathbf{X} = \{X_{i,j}\}$:

$$X_{i,j} = \sum_{r=0}^{7} \sum_{s=0}^{7} \left\{ \frac{w_r w_s}{4} \cdot \cos \frac{\pi}{16} r(2i+1) \cdot \cos \frac{\pi}{16} s(2j+1) \cdot (x_{r,s} - 128) \right\}, \tag{1}$$

where

$$w_r = \begin{cases} \frac{1}{\sqrt{2}}, & r = 0 \\ 1, & r > 0 \end{cases}. \tag{2}$$

The coefficient $X_{0,0}$ is called the DC component of the associated block; the remaining $X_{i,j}$s are called AC components.

The resulting DCT coefficients are then quantized with a predetermined quantization table $\mathbf{Q} = \{Q_{i,j}\}$, i.e.,

$$\tilde{X}_{i,j} = \text{round}\left(\frac{X_{i,j}}{Q_{i,j}}\right), i, j \in \{0, 1, \ldots, 7\}. \tag{3}$$

Most JPEG implementations use a set of quantization tables indexed by a QF ranging from 1 to 100, achieving different rate-distortion (RD) trade-offs. If not otherwise stated, we use the quantization tables recommended in the reference implementation provided by the independent JPEG group [34].

The quantized DCT coefficients of each block are then arranged in a zigzag order. As the DC components of adjacent blocks are usually correlated, the quantized DC components are encoded by using differential pulse code modulation (DPCM). Specifically, the differences of DC components between adjacent blocks are first generated as follows:

$$Dif_{0,0}^{k} = D_{0,0}^{k} - D_{0,0}^{k-1}, \tag{4}$$

Table 1. VLIs' Categories Corresponding to the Coefficient Amplitude

CATEGORIES	COEFFICIENT AMPLITUDE
1	−1,1
2	−3,−2,2,3
3	−7..−4,4..7
4	−15..8,8..15
5	−31..−16,16..31
6	−63..−32,32..63
7	−127..−64,64..127
8	−255..−128,128..255
9	−511..−256,256..511
10	−1023..−512,512..1023
11	−2047..−1024,1024..2047

where $D^k_{0,0}$ is the DC component of the kth block. The difference $Dif^k_{0,0}$ is then encoded into intermediate symbols $(C)(V)$, where C denotes the category of $Dif^k_{0,0}$ (i.e., the number of bits needed to represent it), and V denotes the amplitude of $Dif^k_{0,0}$. For 8-bit images, it can be calculated that C ranges from 0 to 11, as shown in Table 1. Further, C is encoded with a variable-length code (VLC) and V is encoded with a variable length integer (VLI) code whose length in bits is also given in Table 1.

The quantized AC coefficients are pre-encoded using run length encoding (RLE). Each nonzero coefficient is treated as a coding unit and coded into intermediate symbols $(R, C)(V)$. Here, R is the number of consecutive zeros in the zigzag sequence before each nonzero AC coefficient, whose category and amplitude are denoted by C and V, respectively. Then, (R, C) and V are applied with Huffman coding and VLI coding, respectively. Clearly, the coding rate is determined by the locations of these nonzero coefficients and their code lengths.

The JPEG decoding can be performed in the reverse order. Upon receiving the coded JPEG bitstream, we apply the entropy decoding to obtain the quantized DCT blocks. We can then generate the dequantized DCT coefficients by

$$\hat{X}_{i,j} = Q_{i,j} \cdot \tilde{X}_{i,j}. \tag{5}$$

Eventually, the decoded image block can be produced by

$$\hat{\mathbf{x}} = \text{trunc}(\text{round}(\text{IDCT}(\hat{\mathbf{X}}))), \tag{6}$$

where $\hat{\mathbf{X}} = \{\hat{X}_{i,j}\}$, IDCT$(\cdot)$ represents the inverse DCT, and trunc$(\cdot)$ denotes the truncation function such that the resulting pixel values are within the range of $[0, 255]$. The two functions round$(\cdot)$ and trunc$(\cdot)$ are applied elementwise.

3 HOW DOES FACEBOOK MANIPULATE UPLOADED IMAGES?

It has been observed that almost all the existing OSNs apply various operations, e.g., compression and enhancement filtering, to the uploaded images [21, 24] to save storage cost or improve viewing experiences. Typically, these operations are lossy in nature and could severely affect the privacy-protection strategy applied to the uploaded images. Therefore, to design a viable privacy-preserving image-sharing scheme robust against these lossy operations, we should first know how OSNs manipulate uploaded images. As mentioned earlier, our primary focus in this work is on

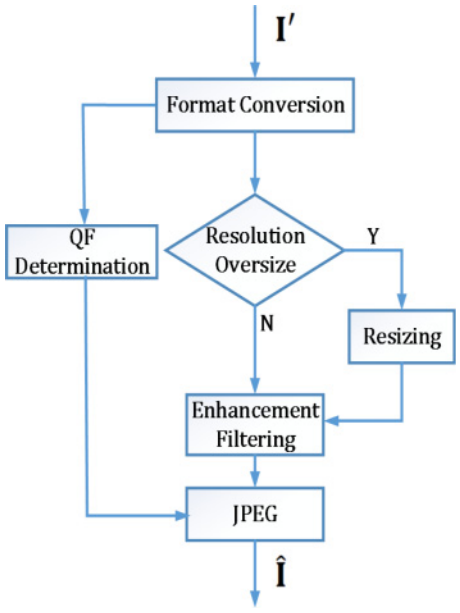


Fig. 2. Manipulations of uploaded images on Facebook.

Facebook because (1) Facebook is one of the most popular OSNs that integrates image sharing; and (2) according to [21] and our observations, the operations conducted by Facebook are much more complex than the other OSNs, such as Twitter and WeChat. The results obtained over Facebook could be easily extended to the other platforms. It should be noted that, although [21] investigated this problem, some of their results are not accurate and many are no longer valid, probably due to the system update of Facebook.

The manipulations that Facebook applies to the uploaded images are illustrated in Figure 2. Specifically, we find that Facebook first converts the images into pixel domain regardless of their original formats. For instance, if the image is JPEG compressed, Facebook first decodes it into pixel values and truncates them to the range [0, 255] (corresponding to 8-bit). This is reasonable as the subsequent JPEG compression, resolution check, etc., are all conducted in the pixel domain. In addition, as will be shown shortly, pixel value truncation during format conversion could lead to a severe pixel overflow problem, which has to be tackled to achieve desirable robustness.

Facebook then applies the following three types of operations adaptively according to image characteristics: (1) resizing, (2) JPEG compression, and (3) enhancement filtering.

Resizing: It is observed that Facebook performs resizing when the resolution of the uploaded image is too large. As resizing could cause significant information loss, it is crucial to limit the resolution of the uploaded image such that no resizing is triggered; otherwise, the information loss is difficult to recover.

To know the maximum tolerable resolution, we upload a series of images of different resolutions to Facebook and compare them with the downloaded versions. The results are presented in Table 2. As can be observed, when both the height and width are less than 2048 pixels, the image resolution remains intact. Otherwise, resizing is conducted.

JPEG Compression: All uploaded images are applied to a round of JPEG compression by Facebook, during which the users *cannot* choose the values of QF. As the distortion caused by JPEG

Table 2. Image Resizing Test on Facebook

Original	Downloaded	Original	Downloaded	Original	Downloaded
128×128	128×128	1024×1024	1024×1024	1500×1800	1500×1800
256×256	256×256	960×1500	960×1500	1800×1800	1800×1800
512×512	512×512	1024×1500	1024×1500	2408×2048	2048×2048
960×960	960×960	1500×1500	1500×1500	2163×2209	2005×2048
960×1024	960×1024	1024×1800	1024×1800	2500×2049	2048×1679

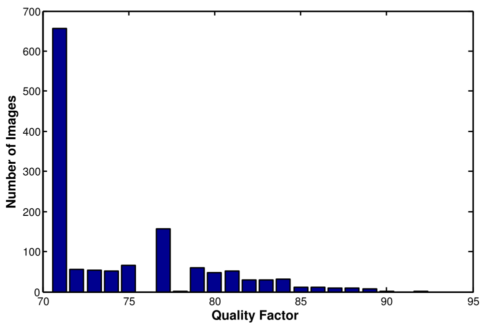


Fig. 3. Distribution of the employed QFs.

compression is directly related to the employed QF, it is of great importance to know the mechanism of assigning QFs for different images. Prior work [21] only roughly studied this issue and claimed that the employed QFs are within the range of [76, 86].

To gain more accurate and in-depth understanding, we upload all 1338 images from UCID-v2 [26] to Facebook and then extract the quantization tables from the downloaded images. It is observed that the quantization tables match exactly with the ones included in the International JPEG Group standard [34]. However, the employed QF values are image dependent and range from 71 to 92, as can be seen from Figure 3. Generally, for images with a large amount of activities, the QF values tend to be low, while for those with large portion of homogenous regions, the QF values seem to be high. This can be more effective to limit the sizes of the compressed files. We also have another very important observation: for encrypted images with randomized structures, no matter which type of reasonably designed encryption algorithm is applied, the QFs adopted are *consistently* 71. Essentially, Facebook treats encrypted images as high-activity images and applies the lowest QF to maximally reduce the file sizes.

Enhancement Filtering: In addition to JPEG compression, Facebook applies another level of enhancement filtering to improve the appearance of the images. We find that it is very challenging to exactly determine such enhancement filtering, as it is applied locally and is highly adaptive. The designed privacy-preserving image-sharing scheme should be robust to these *unknown* lossy operations as well.

4 SYSTEM MODEL AND DESIGN GOALS

Being aware of the lossy operations to be conducted by Facebook, we aim to design a robust privacy-preserving image-sharing scheme. An overview of the whole system is depicted in Figure 4. A user Alice first encrypts the original image $\mathbf{I}$ into $\mathbf{I}'$ by using an encryption function $\mathtt{Encry}$, i.e.,

$$\mathbf{I}' = \mathtt{Encry}(\mathbf{I}, \mathbf{K}), \tag{7}$$

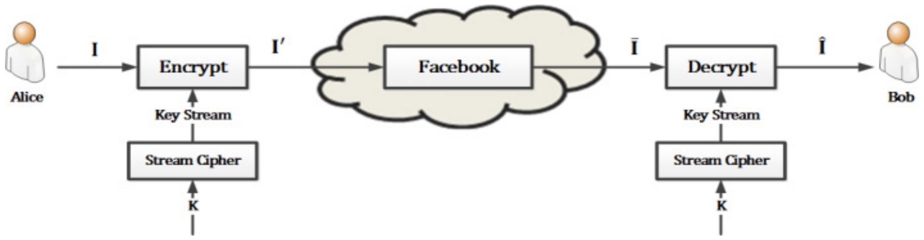


Fig. 4. Overview of the proposed privacy-preserving image sharing system.

where $\mathbf{K}$ is a binary keystream generated from a stream cipher with secret key K[2]. We assume that K has been prenegotiated between Alice and the recipient Bob by using some existing key distribution and management protocols [3, 4, 11, 19]. The length of K is set to be 256 bits. The issues regarding secret key distribution and management are very important as well as challenging in the context of OSNs. So far, there are some works (e.g., based on public key infrastructure) that have been devoted to the design of key distribution and management protocols for OSNs. Interested readers can check the relevant references [3, 4, 11, 19]. As the current work is focused on the end-to-end encryption/decryption schemes that are robust against the lossy operations conducted by OSNs, thorough discussions on secret key distribution and management are beyond the scope of this article.

In addition, we enforce that both the height and the width of $\mathbf{I}$ are no larger than 2048 pixels and, hence, will not lead to resizing operations on Facebook. In practice, when either dimension is larger than 2048 pixels, we recommend applying a user-side downsampling such that the resolution will not trigger resizing.

Alice then uploads the encrypted $\mathbf{I}'$ to Facebook for sharing purposes. As presented in Section 3, Facebook converts the received $\mathbf{I}'$ to the pixel domain and then applies resizing, JPEG compression, and enhancement filtering. Also, we know that the employed QF for JPEG compression is 71. The processed image $\bar{\mathbf{I}}$ of JPEG format is stored in the Facebook server and shared over the Internet. In this sense, Facebook acts as a transcoder such that, for any given input images, the resulting format is always JPEG.

At the client side, Bob downloads $\bar{\mathbf{I}}$ from Facebook and applies the following decryption function to produce an estimated $\hat{\mathbf{I}}$.

$$\hat{\mathbf{I}} = \mathtt{Decry}(\bar{\mathbf{I}}, \mathbf{K}), \tag{8}$$

where $\mathbf{K}$ can be reproduced by using the same key K as Alice does.

The key components of our privacy-preserving image -sharing scheme are the encryption and decryption functions `Encry` and `Decry`. The design goals can be summarized as follows.

(i) ***Reasonably high level of security***: Those parties that do not access the secret key K cannot get meaningful semantic information about the original image. This is the fundamental requirement for any privacy-preserving image-sharing scheme. Similar to many privacy-preserving image-sharing schemes, e.g., [24, 38, 39], we are not targeting data security in the strict cryptographic sense but rather the aforementioned perceptual security, as advocated by many researchers in the multimedia security field.

(ii) ***High quality of the reconstructed image***: The reconstructed image $\hat{\mathbf{I}}$ should be high quality. In other words, we aim to minimize end-to-end distortion $d = ||\mathbf{I} - \hat{\mathbf{I}}||_2^2$.

[2]In traditional stream cipher–based encryption schemes, the input data is directly XORed with the generated keystream $\mathbf{K}$. In our proposed scheme, the way of generating the keystream $\mathbf{K}$ is the same but how to use it is very different.

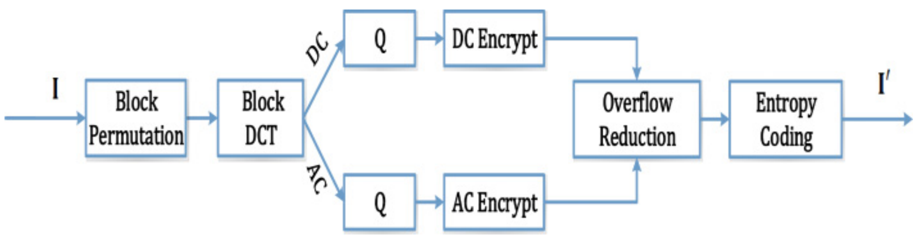


Fig. 5. Proposed image encryption scheme.

(iii) ***High efficiency of storage on Facebook***: We also would like to minimize the storage cost on Facebook, as the number of images currently shared has become astonishingly large. This is also related to the bandwidth consumption of both Alice and Bob when uploading/downloading images to/from Facebook.

To meet the above design goals, a big challenge we face is how to combat the negative effect brought by the lossy operations on Facebook, which are applied automatically and cannot be controlled by users.

5 PROPOSED IMAGE ENCRYPTION AND DECRYPTION

Aided by knowledge of lossy operations conducted by Facebook, we now design the key components of the proposed privacy-preserving image-sharing scheme illustrated in Figure 4: image encryption `Encry` and decryption `Decry` modules. Bear in mind that these two modules should be designed in a way to meet the above goals. Since the dominating degradation source on Facebook is JPEG compression carried out over the DCT domain, it is natural to choose the DCT domain to perform encryption to better limit distortions incurred. In addition, randomization caused by DCT-domain encryption would inevitably generate severe pixel overflow upon being converted to the pixel domain, which has to be tackled to ensure high quality of the reconstructed images.

Motivated by the above intuition, we present the design of our proposed image encryption scheme in Figure 5. The incoming image $\mathbf{I}$ is partitioned into a series of nonoverlapping blocks of size 8×8, which matches the block size used for JPEG compression on Facebook. We then treat each 8×8 pixel block as a processing unit and apply a block-based permutation to destroy the interblock correlation of the input image $\mathbf{I}$. Let $\mathbf{x} = \{x_{r,s}\}$ be a generic 8×8 pixel block upon the block-based permutation. We apply DCT to each $\mathbf{x}$ and obtain the corresponding DCT coefficient block $\mathbf{X} = \{X_{i,j}\}$. As we know that the encrypted images will be subject to JPEG compression with $QF = 71$ on Facebook, we quantize each DCT coefficient block $\mathbf{X}$ by using the JPEG quantization table associated with $QF = 71$. It can be easily seen that this strategy of choosing the quantization table could minimize end-to-end distortion, which satisfies our second design goal. Due to the distinct characteristics of the quantized DC and AC coefficients, we employ different encryption strategies that are discussed separately below.

5.1 Encryption of DC Coefficients

It is well known that a significant fraction of total image energy is concentrated on the DC parts. However, the percentage of bits used for encoding DC coefficients is relatively low, typically around 10% when $QF = 71$[3]. This motivates us to pay more attention to the robustness of the DC encryption scheme to ensure the high-quality reconstruction of DC parts while allowing a

[3]We perform the test over 100 images and find that DC and AC parts contribute 10.57% and 89.43% to the final coded bits, respectively.

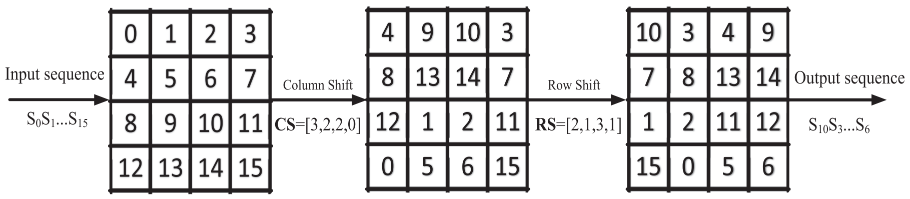


Fig. 6. An example of the cyclical shifts.

relatively large increase in their coding rates. JPEG encodes DC coefficients through a DPCM mechanism, which exploits the interblock correlation of natural images. We propose encrypting DC coefficients by applying permutations among them, only changing their positions but preserving their values. This encryption scheme has several advantages: (1) JPEG compression conducted by Facebook will not enlarge the distortion in the DC parts—this is because permutation does not change the DC coefficients values, which are all quantized with the same quantization parameter corresponding to $QF = 71$; and (2) permutation does not fully destroy interblock correlation, which could benefit the encoding of the DC coefficients.

Specifically, we can easily get the quantized DC coefficients of all the blocks of the input image **I**, and form a DC vector

$$\mathbf{DC} = (DC_0, DC_1, \ldots, DC_{n-1})', \tag{9}$$

where n is the number of blocks in **I**. We reshape the vector **DC** into a 2-D block having four columns and $\lceil n/4 \rceil$ rows. We then perform two key-driven cyclical shift operations to the resulting 2-D block of DC components and read out the data in raster-scan order to obtain the permuted DC vector

$$\tilde{\mathbf{DC}} = (\tilde{DC}_0, \tilde{DC}_1, \ldots, \tilde{DC}_{n-1})'. \tag{10}$$

Let **CS** and **RS** be the secret key vectors controlling the column and the row shift offsets for **DC**. Here, **CS** and **RS** are extracted from the keystream **K** generated by a stream cipher. This implies that the employed key vectors could be different, even for the same image encrypted at different sessions. The random permutation can also be illustrated in Figure 6 for an input sequence $S = s_0 s_1 \cdots s_{15}$, where the numbers within the blocks denote the indexes of the elements of S. Before permutation, the first row becomes $(0, 1, 2, 3)$, the second row becomes $(4, 5, 6, 7)$, on so on. The column shifts are specified by a key vector $\mathbf{CS} = [3, 2, 2, 0]$, with each column undergoing a downward cyclical shift in accordance with the key value associated with that column. The procedure is then repeated using another key vector $\mathbf{RS} = [2, 1, 3, 1]$ for each of the rows. Note that such permutation operations can be realized via circular shifts, which can be easily implemented in either hardware or software.

5.2 Encryption of AC Coefficients

Compared with DC coefficients, AC coefficients focus more on representing the high-frequency details and generally occupy 90% of the final compressed bitstream, when $QF = 71$. Hence, when designing the encryption method for AC parts, we should pay more attention to minimizing the effect on the coding rate. In JPEG compression, AC coefficients are encoded by run-length coding, implying that the coding rate is determined by the locations of nonzero coefficients and their code lengths. To minimize the storage overhead of the encrypted images (the third design goal), our strategy of encrypting the AC coefficients is to XOR the nonzero coefficients with a keystream from a stream cipher while maintaining the location and binary length of each nonzero AC coefficient unaltered. In addition, we keep all zero AC coefficients unchanged. The encryption of the AC

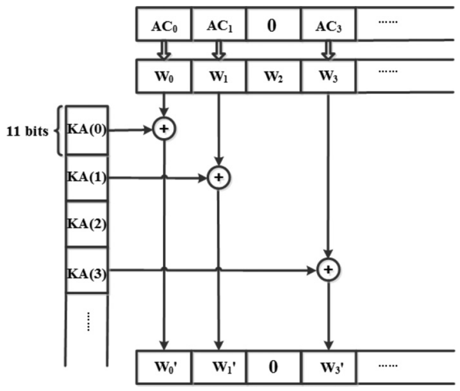


Fig. 7. Encryption of the AC components.

coefficients is illustrated in Figure 7 and the steps of performing the encryption are detailed as follows.

Step 1. Let $\mathbf{KA}$ be a subset of the keystream $\mathbf{K}$ responsible for the encryption of AC coefficients. Partition $\mathbf{KA}$ into $\mathbf{KA} = \mathbf{KA}(0), \mathbf{KA}(1), \ldots,$ where each $\mathbf{KA}(i)$ is of length 11 bits.

Step 2. Convert each nonzero AC coefficient AC_i into a bit sequence $\mathbf{W}_i$ of length l_i.

Here, the way of converting the bit sequence is the same as that done by JPEG standard: that is, each coefficient is represented by its ones' complement code without the sign bit. For instance, 7 is converted to bit sequence 111, while -7 is converted to 000.

Step 3. Encrypt AC_i by bitwise XORing $\mathbf{W}_i$ with the first l_i bits of $\mathbf{KA}(i)$, i.e.,

$$\mathbf{W}'_i = \mathbf{W}_i \oplus \mathbf{KA}_{1 \rightharpoonup l_i}(i), \tag{11}$$

where $\mathbf{W}'_i$ is the encrypted version of $\mathbf{W}_i$ and $\mathbf{KA}_{1 \rightharpoonup l_i}(i)$ represents the first l_i bits of $\mathbf{KA}(i)$. The reason why we assign 11 bits for each $\mathbf{KA}(i)$ is to achieve better robustness. The lossy operations on Facebook could make some of the zero AC coefficients be nonzero or may even change the length of some nonzero AC coefficients. To address the challenging issue of achieving synchronization between the encoder and decoder, we associate each AC coefficient, regardless of whether it is zero or nonzero, with a keystream $\mathbf{KA}(i)$ of length 11 bits, which is longer than the maximum length (10 bits) of any AC coefficient [34]. It can be easily seen that the local desynchronization only affects the decryption/decoding of the local AC coefficient; but this error will not propagate to influence subsequent AC coefficients.

Step 4. Convert each binary sequence $\mathbf{W}'_i$ into a signed decimal number, denoted as AC'_i.

Remark. A desirable property of the proposed AC encryption scheme is that the positions and lengths of the nonzero AC coefficients are kept intact. This property makes the encrypted file readily compressible by Facebook without the need of accessing the secret key K.

5.3 Pixel Overflow Reduction

As discussed in Section 3 (see Figure 2), Facebook converts the uploaded images to the pixel domain and performs truncation for overflowed pixels prior to applying JPEG compression. Due to the randomization incurred by DCT-domain encryption, we observe severe pixel overflow upon the format conversion. That is, many converted pixel values will be outside of the range $[0, 255]$ and will be automatically modified to 0 or 255, causing large distortions in the reconstructed images. It should be noted that most of the existing DCT-domain encryption schemes, e.g., [37, 38], did not address this challenging issue, because the encrypted files stay in the DCT domain. However, in the current scenario, we cannot control the format conversion performed by Facebook and, hence, have to deal with this challenge. The most relevant existing work regarding this point is found in [9]. As will be clear soon, our proposed strategy is a much generalized solution and the associated parameters are determined in an optimal sense, rather than from heuristics.

In the following, we propose an active solution by incorporating DCT coefficient shrinkage into our encryption phase. Specifically, after applying the DCT-domain encryption as described in Section 5.1 and 5.2, we mimic the operations of Facebook by performing IDCT to encrypted DCT blocks. We then record the blocks that contain the overflow pixels in a location map $\mathbf{M} = \{M_{i,j}\}$, namely,

$$M_{i,j} = \begin{cases} 1, & (i,j)\text{th block contains overflow pixels} \\ 0, & \text{otherwise} \end{cases}. \tag{12}$$

In general, the location map is sparse and hence can be readily compressed, e.g., by using the JBIG. The compressed **M** can be uploaded in plaintext to Facebook as metadata.

Let $\mathbf{T} = \{T_{i,j}\}$ be a generic DCT coefficient block having the overflow problem. Our strategy of resolving the pixel overflow problem is to reduce block energy by multiplying each $T_{i,j}$ with a shrinkage factor $R_{i,j} \in [0,1]$, that is,

$$\mathbf{T}' = \mathbf{T} \odot \mathbf{R}, \tag{13}$$

where $\odot$ is the operator for Hadamard product and $\mathbf{R} = \{R_{i,j}\}$ is the shrinkage matrix. To reduce the overhead of introducing **R**, we apply it to *all* overflowed DCT coefficient blocks. Now, the critical issue is how to determine **R** in a certain optimal sense. As can be easily seen, smaller $R_{i,j}$ can more effectively drag the resulting pixels away from 255 but, at the same time, could kill small AC coefficients to zeros, incurring another type of distortion. Here, we propose an optimization framework to obtain the optimal **R** to minimize the end-to-end distortion of the reconstructed image at the recipient side.

Let $\mathbf{t} = \{t_{i,j}\}$ be the pixel block corresponding to the DCT coefficient block **T** (before shrinkage), i.e.,

$$\mathbf{t} = \text{IDCT}(\mathbf{T}). \tag{14}$$

Upon applying the shrinkage operation on **T**, as shown in Equation (13), and performing pixel truncation to the resulting pixel block, we obtain that

$$\mathbf{t}' = \text{trunc}(\text{round}(\text{IDCT}(\mathbf{T}'))), \tag{15}$$

where the truncation function $y = \text{trunc}(x)$ can be explicitly expressed as

$$y = \begin{cases} 0 & \text{if } x < 0, \\ 255 & \text{if } x > 255, \\ x & \text{otherwise.} \end{cases} \tag{16}$$

We also can mimic the operation in the recipient side by applying the deshrinkage and eventually get a reconstructed pixel block $\hat{\mathbf{t}} = \{\hat{t}_{i,j}\}$, where

$$\hat{t}_{i,j} = \text{round}\left(\frac{t'_{i,j}}{R_{i,j}}\right). \tag{17}$$

Therefore, the end-to-end distortion for a generic DCT block **T** can be written as

$$d = \|\mathbf{t} - \hat{\mathbf{t}}\|_2^2, \tag{18}$$

which is measured over the pixel domain.

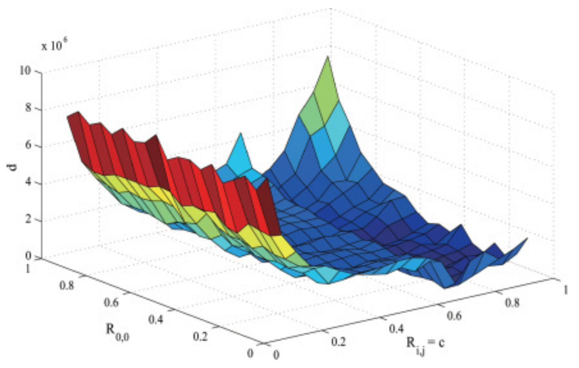


Fig. 8. End-to-end distortion with respect to shrinkage factors.

To determine the shrinkage matrix **R**, we collect 100 training images of size 512×512, apply our proposed DCT-domain encryption, and then detect the problematic blocks with pixel overflow. Define a set Ω consisting of all these problematic blocks. Finally, the determination of **R** is formulated as the following optimization problem to minimize the average end-to-end distortion, i.e.,

$$\hat{\mathbf{R}} = \arg\min_{\mathbf{R}} \frac{1}{|\Omega|} \sum_{\{k|\mathbf{T}_k \in \Omega\}} d_k, \tag{19}$$

where $|\Omega|$ denotes the cardinality of Ω and d_k is the distortion associated with block $\mathbf{T}_k$ and can be similarly calculated as Equation (18).

However, due to the nonlinearities introduced by truncation and rounding, the above optimization problem is nonconvex. To solve this challenging optimization problem, we here adopt a sampling-based offline training method. To reduce the search range, we assume that $R_{i,j} = c$ for all $(i,j) \neq (0,0)$, where c is an unknown to be estimated. The relationship between the average distortion with respect to $R_{0,0}$ and c is illustrated in Figure 8. As can be observed, there are many local minima of this cost function, suggesting that it is indeed nonconvex. The optimization problem now becomes to find $R_{0,0}$ and c to minimize the above cost function in Equation (19). We then traverse all possible $R_{0,0} \in [0,1]$ and $c \in [0,1]$ with step size 0.05, attempting to find the pair $(R_{0,0}, c)$ that gives the lowest value of the average distortion. In total, we need to check 400 combinations of different $(R_{0,0}, c)$. Based on this offline training process, we choose the shrinkage factors as

$$R_{i,j} = \begin{cases} 0.3, & (i,j) = (0,0) \\ 0.75, & \text{otherwise} \end{cases}. \tag{20}$$

The whole offline training process takes around 6 hours. The measurement of the training time complexity is carried out over an unoptimized, unparalleled MATLAB implementation by using the built-in tic and toc functions in a personal PC with Intel i7 with a 3.40GHz CPU and 8GB RAM. We need to perform the above training process only once, and the obtained factors will be applied to all problematic blocks.

Our proposed technique for handling the overflow pixel can be regarded as an extension to the technique developed in [9], in which an empirically determined factor 0.5 is applied to all DCT coefficients. In contrast, we only apply the shrinkage to those problematic blocks, better limiting the incurred distortions. More important, the determination of the shrinkage matrix is conducted within an optimization framework instead of from heuristics.

It should also be noted that the above shrinkage strategy has another side effect: reducing the bit rate. This is because some of the small AC coefficients are killed to zeros, potentially increasing the run length. In addition, the amplitudes of the AC coefficients are reduced, making it more efficient to be coded.

As will be demonstrated in our experimental results, the pixel overflow reduction solution presented above can significantly improve the quality of the reconstructed images at the recipient side.

5.4 Decryption Function Decry

Upon downloading the encrypted JPEG file from Facebook, recipient Bob aims to decrypt and decode it into an image in the pixel domain with the assistance of the secret key K and the location map $\mathbf{M}$. Essentially, the decryption is the inverse of the encryption. The JPEG file can be readily partitioned into segments, each of which corresponds to the bitstream for an 8×8 pixel block. For each bitstream segment, we apply entropy decoding and dequantization and obtain the corresponding DCT block. We then check the location map $\mathbf{M}$, in which we record the overflow blocks. For each overflow block, the associated DC and AC coefficients are subject to deshrinkage operations by dividing the shrinkage factors as shown in Equation (17).

After that, we can get all encrypted DC coefficients and form a permuted DC vector $\hat{\mathbf{DC}}$. To restore the original DC vector, we apply a 2-D permutation, with column and row shift vectors matched with $\mathbf{CS}$ and $\mathbf{RS}$ used at the encryption stage.

To decrypt the AC coefficients, we apply the following operations:

Step 1. Let $\mathbf{KA}$ be the keystream used in the encryption phase, which can be generated by using the same secret key K at the recipient side. Partition $\mathbf{KA}$ into $\mathbf{KA} = \mathbf{KA}(0), \mathbf{KA}(1), \ldots$, where each $\mathbf{KA}(i)$ is of length 11 bits.

Step 2. Convert each nonzero AC coefficient $\hat{AC}_i'$ into a bit sequence $\hat{\mathbf{W}}_i'$ of length l_i.

Step 3. Decrypt $\hat{AC}_i'$ by bitwise XORing $\mathbf{W}_i'$ with the first l_i bits of $\mathbf{KA}(i)$, i.e.,

$$\hat{\mathbf{W}}_i = \hat{\mathbf{W}}_i' \oplus \mathbf{KA}_{1 \rightharpoonup l_i}(i), \tag{21}$$

where $\hat{\mathbf{W}}_i$ is the decrypted version of $\hat{\mathbf{W}}_i'$ and $\mathbf{KA}_{1 \rightharpoonup l_i}(i)$ represents the first l_i bits of $\mathbf{KA}(i)$.

Step 4. Convert each binary sequence $\hat{\mathbf{W}}_i$ into a signed decimal number, denoted as $\hat{AC}_i$.

After decrypting all DC and AC components, the corresponding pixel blocks can be easily recovered by applying the IDCT. Finally, the recovered image can be produced by applying a round of pixel block permutation.

We conclude this section by giving a short remark on the extensibility of our proposed scheme to the other OSN platforms. In fact, the proposed robust privacy-preserving image-sharing scheme over Facebook can be readily extended to other platforms, such as Twitter and WeChat. For these two platforms, we also observe that consistent QF values are applied to encrypted images. Specifically, for Twitter, the QF value employed is 71; for WeChat, this number increases to 77. With proper modifications on some system parameters (e.g., the QF value and the retrained shrinkage factors), a similar system can be designed to achieve privacy protection of shared images over these OSNs.

6 SECURITY ANALYSIS

In this section, we present an analysis of the security of the proposed privacy-preserving image-sharing scheme.

Recall that the keystream controlling the pixel blocks permutation, DC vector permutation, and AC coefficient encryption is generated by using a stream cipher. This implies that the keystream is different for each image and it could be different even for the same image encrypted at different sessions. In this case, the Chosen-Plaintext Attack (CPA) or Chosen-Ciphertext Attack (CCA) cannot get any useful information—only the expired keystreams at most. Hence, the only attack model applicable to our proposed privacy-preserving image-sharing scheme is the Ciphertext-Only Attack (COA), in which the attacker can access only the encrypted JPEG file and attempts to recover the original image.

As the entropy coding part is completely public and invertible, the attacker can obtain the encrypted (quantized) DC vector $\tilde{\mathbf{DC}} = (\tilde{DC}_0, \tilde{DC}_1, \ldots, \tilde{DC}_{n-1})$ and AC coefficients. With $\tilde{\mathbf{DC}}$, the following statistical attack could be applied. Let $\mathcal{D} = \{D_0, D_1, \ldots, D_{m-1}\}$ be the set consisting of all the *distinct* values of $\tilde{\mathbf{DC}}$, where $m \leq n$. For each D_i, we calculate its empirical probability mass function (EPMF) by

$$p_i = \frac{\#D_i}{n}, \tag{22}$$

where $\#D_i$ denotes the number of D_i in the vector $\tilde{\mathbf{DC}}$. The following entropy quantity can then be used to measure the complexity of the input image:

$$h = -\sum_{i=0}^{m-1} p_i \log_2 p_i \tag{23}$$

Clearly, images with intensive fine details would result in larger values of h, while images with a large portion of smooth regions would give smaller values. In other words, some statistical information of the input image leaks. This situation is similar to that faced by any permutation-based image cipher [45]. However, for general consumer electronics, including social media applications, leakage of statistical information seems to be tolerable or even inevitable [24, 38, 39, 45].

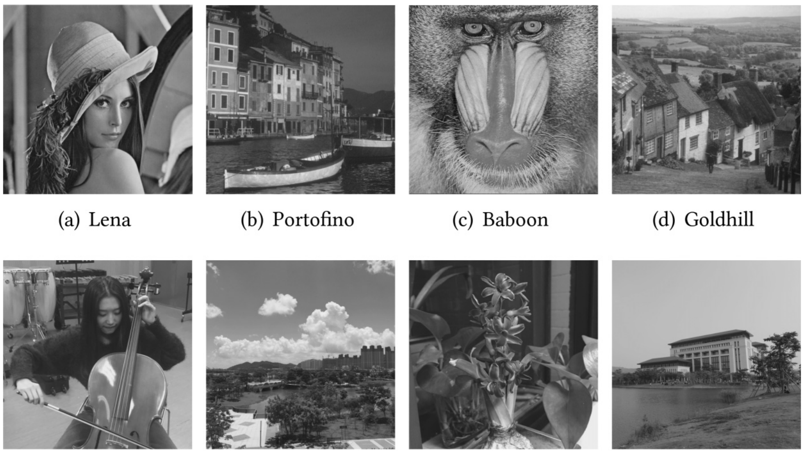
Despite statistical information leakage, it is practically intractable to figure out the permutation, due to the large number of *distinct* ways of performing permutations. Specifically, the number of distinct ways of permutation for the DC vector can be computed by

$$\frac{n!}{\Pi_{i=0}^{m-1}(\lfloor p_i \cdot n \rfloor !)}. \tag{24}$$

In practice, the number given by Equation (24) is extremely large, precluding practical brute-force attack. For instance, the number of distinct ways of permutation of the DC vector for the Lena image is significantly larger than 2^{256}, which is the size of the key space.

For the AC coefficients, important information that an attacker can exploit is that zero AC coefficients remain unchanged before and after encryption. Though some of the zero (nonzero) AC coefficients are likely to be changed to nonzero (zero) ones due to the lossy operations conducted on Facebook, we observe that the occurrence probability is rather low. Hence, a good strategy for an attacker is to estimate only those nonzero AC coefficients while leaving the zero AC coefficients intact. The nonzero AC coefficients are encrypted by the generated keystream via XOR operations. One possible attack strategy for AC coefficients is to estimate the signs of nonzero AC coefficients first and then assign constant amplitudes, as practiced in [35].

Based upon the above analysis, an attacker can randomly guess a permutation way of the DC vector and accordingly obtain the estimated DC vector. For the AC coefficients, an attacker simply keeps all the zero AC coefficients unchanged and randomly selects the signs of the AC coefficients following a certain distribution. Here, as a favorable condition for an attacker, we assume that the distribution of the AC coefficient signs is available to the attacker. As will be shown experimentally in the next section, even under this favorable assumption, an attacker can only restore images with very poor qualities.


(a) Lena (b) Portofino (c) Baboon (d) Goldhill

(e) Cellist (f) Summer (g) Hyacinth (h) Building

Fig. 9. Some representative test images.

7 EXPERIMENTAL RESULTS

In this section, the security and performance of our proposed privacy-preserving image-sharing scheme are evaluated experimentally. The test set is composed of 100 images of sizes 512×512, with various characteristics including portrait images, landscape images, and self-taken photos. Some representative test images are seen in Figure 9. Our experiments are conducted on MATLAB 2013b using a personal PC with Intel i7 with a 3.40GHz CPU and 8GB RAM. It takes around 1 second, on average, to process one image. All of the encrypted images are uploaded to Facebook and then downloaded. The decryption and reconstruction are implemented in the same experimental environment. Thus, all of the following results are based on real Facebook data.

7.1 Evaluation on Security

We first demonstrate that our proposed privacy-preserving image-sharing scheme can effectively destroy the semantic meaning of the original images. In Figure 10, we show the encrypted versions ($\mathbf{I}'$ in Figure 4) of the test image `Lena` and `Cellist`. To further show the rationality behind the proposed scheme, we also give some intermediate results by only applying the pixel block permutation or by removing the pixel block permutation while maintaining the subsequent DC and AC scrambling. As can be seen from Figure 10(b),(f), when only the pixel block permutation is conducted, the content in each block is well preserved, revealing significant information of the original image. For instance, the eyes of the `Lena` image can still be detected. Meanwhile, if we remove the pixel block permutation, as illustrated in Figure 10(c),(g), we can roughly see the contour of the original image. This is because, when we perform the AC coefficients encryption, we intentionally do not change the zero AC coefficients. However, when the pixel block permutation and the DC/AC scrambling work together, the semantic meaning of the original images has been effectively destroyed.

Furthermore, we compare with some state-of-the-art privacy-preserving image-sharing schemes—P3 [24], Cryptagram [32], and Secure JPEG [38]—in terms of the visual quality of the encrypted image $\mathbf{I}'$. For fair comparison, we assume that all these competing methods know the employed $QF = 71$ on Facebook, which is certainly an advantage for them. As can be observed from Figure 11, Facebook can hardly get any meaningful information from the images encrypted

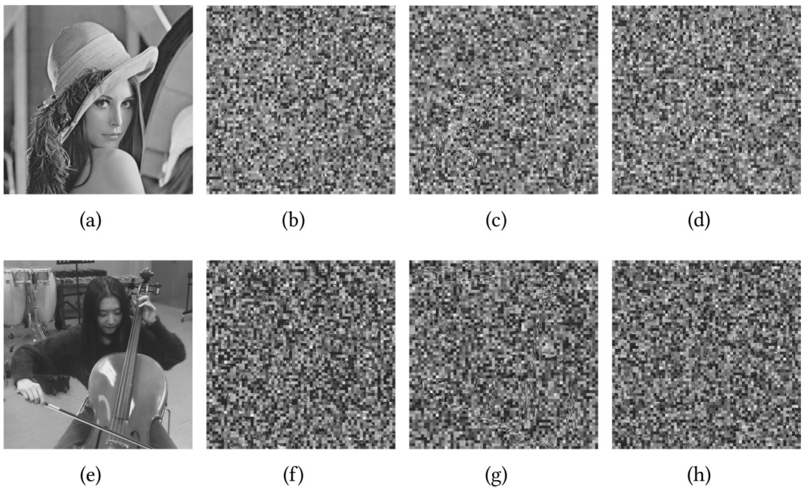


Fig. 10. (a) Original Lena. (b) Encrypted Lena by applying pixel block operations only. (c) Encrypted Lena by performing DC/AC scrambling only. (d) Encrypted Lena by our proposed scheme. (e) Original Cellist. (f) Encrypted Cellist by applying pixel block operations only. (g) Encrypted Cellist by performing DC/AC scrambling only. (h) Encrypted Cellist by our proposed scheme.

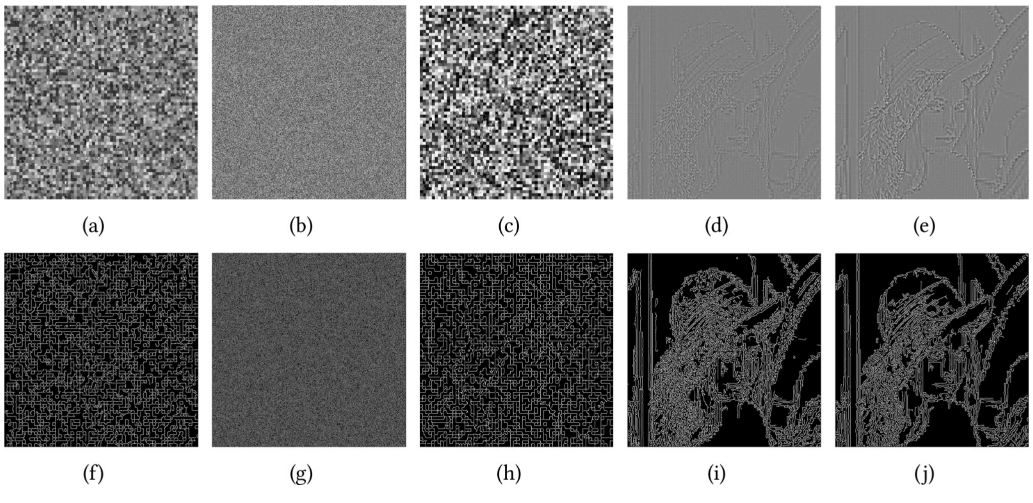
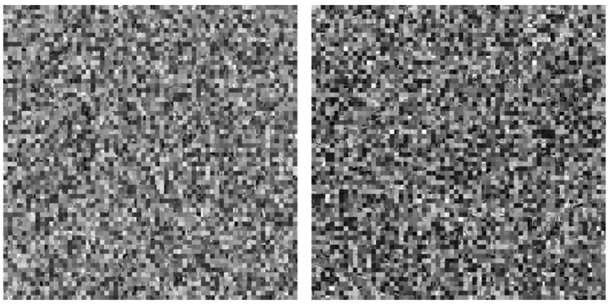


Fig. 11. Visual Comparison of Encrypted Lena Images and their edge detected versions. (a),(f) Proposed; (b),(g) Cryptagram; (c),(h) Secure JPEG; (d),(i) P3 ($T = 10$); and (e),(j) P3 ($T = 20$).

by using our proposed method, even after being enhanced via edge detection. This desirable property holds for Cryptagram and Secure JPEG as well. However, for P3, the public parts available to the attacker reveal significant visual information of the original image. The edge detected version could be treated as a sketch of great fidelity, and the quality is improved with the increasing parameter T [24]. We also have tried some other images, and similar observations can be obtained.

Also, we implement the attack strategy presented in Section 6 and provide the recovered images for Lena and Cellist in Figure 12. It can be observed that both recovered images are of rather



(a) (b)

Fig. 12. Images recovered by the attacker: (a) Lena and (b) `Cellist`.

Table 3. Comparison of the Quality (PSNR in dB) of the Reconstructed Images

	JPEG ($QF = 71$)	Proposed	Secure JPEG	P3 (T=10)	Cryptagram ($Y^3_{2\times2}$)	Cryptagram ($Y^3_{1\times1}$)
`Lena`	37.40	36.85	28.85	36.84	∞	18.47
`Portofino`	35.51	35.18	27.71	35.46	∞	18.40
`Baboon`	30.64	30.27	24.51	30.63	∞	18.74
`Goldhill`	35.27	35.05	29.24	35.24	∞	18.45
`Cellist`	38.65	37.89	27.98	38.41	∞	18.51
`Summer`	35.86	35.46	27.88	35.77	∞	18.34
`Hyacinth`	39.35	38.54	28.06	39.13	∞	17.92
`Building`	37.03	36.66	27.84	36.82	∞	18.52
Average	34.56	33.90	26.62	34.12	∞	18.63

Note: The average is computed over the 100 images in the test set.

poor quality. This indicates that our proposed privacy-preserving image-sharing scheme is secure against the aforementioned attack.

7.2 Evaluation on Reconstruction Quality

We now investigate the quality of the reconstructed images ($\hat{\mathbf{I}}$ in Figure 4) of our proposed privacy-preserving image-sharing scheme and compare it with the ones generated by the state-of-the-art methods. As can be seen from Table 3, the average PSNR of the reconstructed images $\hat{\mathbf{I}}$ (with respect to the original image $\mathbf{I}$) is slightly inferior (0.66 dB worse), when comparing with the offline JPEG coded image with $QF = 71$. Here, the average PSNR values are computed over all 100 test images. The inevitable degradation of the PSNR value is due to the lossy operations conducted by Facebook. Such small degradation indicates that our proposed method can effectively suppress the distortion incurred by the lossy operations on Facebook. In contrast, the Secure JPEG [38] leads to severe PSNR degradation, with the average drop being 7.94 dB. On the other hand, P3 [24] has its inherent advantages in terms of the quality of the reconstructed images because the majority of the information is stored in an error-free cloud server. It should be noted that the slightly higher reconstruction quality of P3 (0.22 db higher than ours) is achieved through the introduction of a third-party cloud server, which may not be realistic and trusted in many situations. Also, as demonstrated previously, the encrypted images generated by P3 leak significant information regarding the original ones. For Cryptagram in $Y^3_{2\times2}$ mode, the reconstructed images are error free; hence, the PSNR values are constantly infinity. As will be shown shortly, this desirable property is achieved at

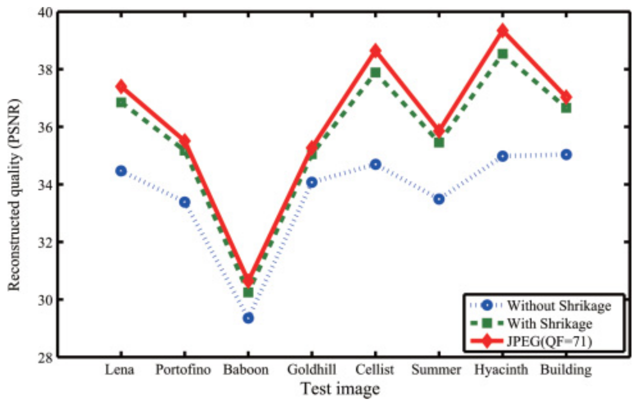


Fig. 13. The effect of applying DCT coefficients shrinkage.

the cost of dramatically increased storage overhead. In addition, when the mode is switched to $Y^3_{1\times1}$, the perfect reconstruction no longer holds. When there are some errors that cannot be corrected (e.g., caused by the lossy operations on Facebook), error propagation arises, which eventually results in severely distorted images. For instance, in $Y^3_{1\times1}$ mode, the average PSNR decreases to 18.63 dB, which is rather low. It should be noted that both $Y^3_{2\times2}$ and $Y^3_{1\times1}$ modes are recommended in [32].

We would also like to further experimentally verify the necessity of handling the pixel overflow problem via DCT shrinkage in our proposed scheme. Figure 13 shows the reconstruction quality of eight test images with/without applying the proposed shrinkage strategy, for which we also give the offline JPEG ($QF = 71$) as reference. As can be seen, the shrinkage technique can achieve PSNR gain as large as 3.55 db, which occurs in the `Hyacinth` image.

7.3 Evaluation on Storage Overhead and Bandwidth Consumption

We also study the storage overhead on Facebook and the bandwidth consumption incurred by different methods. This has become an important evaluation criterion due to the huge number of shared images. The uplink and downlink bandwidths are determined by the file sizes of the encrypted image $\mathbf{I}'$ and the online stored image $\bar{\mathbf{I}}$, respectively (see Figure 4). We observe that the manipulations applied on $\mathbf{I}'$ by Facebook only slightly change its file size, and hence, the uplink bandwidth, the downlink bandwidth, and the online storage cost are very close to each other. Hence, in Table 4, we present only the storage overhead results, in which the overhead is calculated with respect to the offline JPEG coded image with $QF = 71$. On average, our proposed scheme incurs only a 2.0% bit rate increase, which is rather small considering the additional privacy-preserving functionality provided. For P3, the overhead becomes much larger, reaching 16%, on average. We also compare our method with Secure JPEG. For a fair comparison, we choose the ultra-high encryption level, which is applied to the whole image. As can be observed, the average overhead is 5.6%, which is 2.8 times larger than that of our method. For Cryptagram, we give the results for $Y^3_{2\times2}$ and $Y^3_{1\times1}$ modes, as suggested in [32]. When $Y^3_{2\times2}$ mode is adopted, the average storage overhead jumps to 3177%, which is prohibitively large, precluding the practical deployment of this technique. Even when we switch the mode to $Y^3_{1\times1}$, the overhead is still as high as 921%.

From the above analysis, we can see that our proposed method achieves the best trade-off in terms of data privacy, reconstruction quality, and incurred storage overhead.

Table 4. Comparison of Storage Overhead

	Proposed	P3 (T=10)	Secure JPEG	Cryptagram ($Y^3_{2\times2}$)	Cryptagram ($Y^3_{1\times1}$)
Lena	3.2%	20.2%	5.7%	4520%	1310%
Portofino	2.9%	20.0%	5.9%	3797%	1100%
Baboon	1.4%	13.8%	2.2%	2116%	642%
Goldhill	3.7%	15.2%	8.8%	3420%	991%
Cellist	3.5%	19.3%	7.6%	4881%	1411%
Summer	4.3%	15.0%	7.7%	4207%	1217%
Hyacinth	1.9%	18.3%	6.4%	5052%	1461%
Building	6.2%	16.5%	10.0%	5351%	1548%
Average	2.0%	16.0%	5.6%	3177%	921%

Note: The average is computed over the 100 images in the test set.

8 CONCLUSIONS

In this work, we have addressed the problem of designing a privacy-preserving, high-fidelity, and storage-efficient image-sharing scheme over Facebook. Based on a DCT-domain image encryption method robust to various lossy operations conducted by Facebook, we have achieved superior performance in terms of data privacy, quality of the reconstructed images, and storage cost compared with the state-of-the-art competing solutions.

REFERENCES

[1] Bechara Al Bouna, Richard Chbeir, Alban Gabillon, and Patrick Capolsini. 2013. A flexible image-based access control model for social networks. In *Security and Privacy Preserving in Social Networks (Lecture Notes in Social Networks)*. 337–364.

[2] Julian Backes, Michael Backes, Markus Dürmuth, Sebastian Gerling, and Stefan Lorenz. 2011. X-pire!-a digital expiration date for images in social networks. *arXiv Preprint arXiv:1112.2649* (2011).

[3] Ero Balsa, Filipe Beato, and Seda Gürses. 2014. Why can't online social networks encrypt? In *Proceedings of W3C Workshop Privacy UserCentric Controls*.

[4] Filipe Beato, Markulf Kohlweiss, and Karel Wouters. 2011. Scramble! your social network data. In *Proceedings of the International Symposium Privacy Enhancing Technologies Symposium (PETS'11)*. Springer, 211–225.

[5] Andrew Besmer and Heather Richter Lipford. 2008. Privacy perceptions of photo sharing in Facebook. In *Proceedings of the 4th Symposium on Usable Privacy and Security (SOUPS'08)*. ACM.

[6] Bechara Al Bouna, Richard Chbeir, and Alban Gabillon. 2011. The image protector-a flexible security rule specification toolkit. In *Proceedings of the International Conference on Security and Cryptography (SECRYPT'11)*. IEEE, 345–350.

[7] Kwontaeg Choi, Hyeran Byun, and Kar-Ann Toh. 2008. A collaborative face recognition framework on a social network platform. In *Proceedings of the 8th International Conference on Automatic Face and Gesture Recognition (FG'08)*. IEEE, 1–6.

[8] Leucio Antonio Cutillo, Refik Molva, and Melek Önen. 2012. Privacy preserving picture sharing: Enforcing usage control in distributed on-line social networks. In *Proceedings of the 5th Workshop on Social Network Systems (SNS'12)*. ACM, 6.

[9] Ahmet Emir Dirik and Nasir Memon. 2013. Selective robust image encryption for social networks. In *Proceedings of the International Conference Multimedia Communications Services and Security (MCSS'13)*. Springer, 71–81.

[10] Facebook. 2017. Newsroom[Online]. Retrieved December 11, 2017 from http://newsroom.fb.com/company-info/.

[11] Felix Günther, Mark Manulis, and Thorsten Strufe. 2011. Key management in distributed online social networks. In *Proceedings of the IEEE International Symposium on the World of Wireless, Mobile and Multimedia Networks (WoWMoM'11)*. IEEE, 1–7.

[12] Jianping He, Bin Liu, Deguang Kong, Xuan Bao, Na Wang, Hongxia Jin, and George Kesidis. 2016. PUPPIES: Transformation-supported personalized privacy preserving partial image sharing. In *Proceedings of the 46th Annual IEEE/IFIP International Conference on Dependable System Networks (DSN'16)*. IEEE, 359–370.

[13] Zaobo He, Zhipeng Cai, Qilong Han, Weitian Tong, Limin Sun, and Yingshu Li. 2016. An energy efficient privacy-preserving content sharing scheme in mobile social networks. *Personal Ubiquitous Computing* 20, 5, 833–846.

[14] Taiki Honda, Yasutaka Murakami, Yuki Yanagihara, Takeshi Kumaki, and Takeshi Fujino. 2013. Hierarchical image-scrambling method with scramble-level controllability for privacy protection. In *Proceedings of the 56th International Midwest Symposium on Circuits and Systems (MWSCAS'13).* IEEE, 1371–1374.

[15] Panagiotis Ilia, Iasonas Polakis, Elias Athanasopoulos, Federico Maggi, and Sotiris Ioannidis. 2015. Face/Off: Preventing privacy leakage from photos in social networks. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'15).* ACM, 781–792.

[16] Maritza Johnson, Serge Egelman, and Steven M. Bellovin. 2012. Facebook and privacy: It's complicated. In *Proceedings of the 8th Symposium on Usable Privacy and Security (SOUPS'12).* ACM, 9.

[17] Peter Klemperer, Yuan Liang, Michelle Mazurek, Manya Sleeper, Blase Ur, Lujo Bauer, Lorrie Faith Cranor, Nitin Gupta, and Michael Reiter. 2012. Tag, you can see it!: Using tags for access control in photo sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'12).* ACM, 377–386.

[18] Kaitai Liang, Joseph K. Liu, Rongxing Lu, and Duncan S. Wong. 2015. Privacy concerns for photo sharing in online social networks. *IEEE Internet Computing* 19, 2, 58–63.

[19] Ya-Nan Liu, Lein Harn, Lei Mao, and Zhangliang Xiong. 2016. Full-healing group-key distribution in online social networks. *International Journal of Security and Networks* 11, 1–2, 12–24.

[20] Andrew D. Miller and W. Keith Edwards. 2007. Give and take: A study of consumer photo-sharing culture and practice. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'07).* ACM, 347–356.

[21] Jianxia Ning, Inderjit Singh, Harsha V. Madhyastha, Srikanth V. Krishnamurthy, Guohong Cao, and Prasant Mohapatra. 2014. Secret message sharing using online social media. In *Proceedings of the IEEE Conference on Communications and Network Security (CNS'14).* IEEE, 319–327.

[22] Frank Pallas, Max-Robert Ulbricht, Lorena Jaume-Palasí, and Ulrike Höppner. 2014. Offlinetags: A novel privacy approach to online photo sharing. In *Proceedings of the CHI'14 Extended Abstracts Human Factors in Computing Systems (CHI'14).* ACM, 2179–2184.

[23] Iasonas Polakis, Panagiotis Ilia, Federico Maggi, Marco Lancini, Georgios Kontaxis, Stefano Zanero, Sotiris Ioannidis, and Angelos D. Keromytis. 2014. Faces in the distorting mirror: Revisiting photo-based social authentication. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'14).* ACM, 501–512.

[24] Moo-Ryong Ra, Ramesh Govindan, and Antonio Ortega. 2013. P3: Toward privacy-preserving photo sharing. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI'13).* USENIX, 515–528.

[25] David Rosenblum. 2007. What anyone can know: The privacy risks of social networking sites. *IEEE Security and Privacy* 5, 3, 40–49.

[26] Gerald Schaefer and Michal Stich. 2004. UCID - An uncompressed colour image database. *Storage Retrieval Methods and Applications for Multimedia* 5307, 472–480.

[27] Anna Cinzia Squicciarini, Smitha Sundareswaran, Dan Lin, and Josh Wede. 2011. A3p: Adaptive policy prediction for shared images over popular content sharing sites. In *Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia (HT'11).* ACM, 261–270.

[28] Zak Stone, Todd Zickler, and Trevor Darrell. 2008. Autotagging Facebook: Social network context improves photo annotation. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'08).* IEEE, 1–8.

[29] Zak Stone, Todd Zickler, and Trevor Darrell. 2010. Toward large-scale face recognition using social network context. *Proceedings of IEEE* 98, 8, 1408–1415.

[30] Qiudong Sun, Ping Guan, Yongping Qiu, and Yunfeng Xue. 2012. A novel digital image encryption method based on one-dimensional random scrambling. In *Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'10).* IEEE, 1669–1672.

[31] Weiwei Sun, Jiantao Zhou, Ran Lyu, and Shuyuan Zhu. 2016. Processing-aware privacy-preserving photo sharing over online social networks. In *Proceedings of the ACM Multimedia Conference (ACMMM'16).* ACM, 581–585.

[32] Matt Tierney, Ian Spiro, Christoph Bregler, and Lakshminarayanan Subramanian. 2013. Cryptagram: Photo privacy for online social media. In *Proceedings of the ACM Conference on Online Social Networks (COSN'13).* ACM, 75–88.

[33] Lam Tran, Deguang Kong, Hongxia Jin, and Ji Liu. 2016. Privacy-CNH: A framework to detect photo privacy with convolutional neural network using hierarchical features. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'16).* 1317–1323.

[34] Gregory K. Wallace. 1992. The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics* 38, 1, xviii–xxxiv.

[35] Chung-Ping Wu and C.-C. J. Kuo. 2005. Design of integrated multimedia compression and encryption systems. *IEEE Transactions on Multimedia* 7, 5, 828–839.

[36] Kaihe Xu, Yuanxiong Guo, Linke Guo, Yuguang Fang, and Xiaolin Li. 2014. Control of photo sharing over online social networks. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'14).* IEEE, 704–709.

[37] Lin Yuan, Pavel Korshunov, and Touradj Ebrahimi. 2015. Privacy-preserving photo sharing based on a secure JPEG. In *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'15)*. IEEE, 185–190.

[38] Lin Yuan, Pavel Korshunov, and Touradj Ebrahimi. 2015. Secure JPEG scrambling enabling privacy in photo sharing. In *Proceedings of the 11th IEEE International Conference Workshops on Automatic Face and Gesture Recognition (FG'15)*, Vol. 4. IEEE, 1–6.

[39] Lin Yuan, David McNally, Alptekin Küpçü, and Touradj Ebrahimi. 2015. Privacy-preserving photo sharing based on a public key infrastructure. In *Proceedings of Applications of Digital Image Processing XXXVIII (SPIE'15)*. SPIE, 95991I.

[40] Xingliang Yuan, Xinyu Wang, Cong Wang, Anna C. Squicciarini, and Kui Ren. 2016. Towards privacy-preserving and practical image-centric social discovery. *IEEE Transactions on Dependable and Secure Computing*. 1–14.

[41] Zephoria. 2017. The top 20 valuable Facebook statistics. Retrieved December 11, 2017 from https://zephoria.com/top-15-valuable-facebook-statistics/.

[42] Chi Zhang, Jinyuan Sun, Xiaoyan Zhu, and Yuguang Fang. 2010. Privacy and security for online social networks: Challenges and opportunities. *IEEE Network* 24, 4, 13–18.

[43] Lan Zhang, Taeho Jung, Cihang Liu, Xuan Ding, Xiang-Yang Li, and Yunhao Liu. 2014. Outsource photo sharing and searching for mobile devices with privacy protection. *arXiv Preprint arXiv:1410.6589.*

[44] Li Zhang, Xiaolin Tian, and Shaowei Xia. 2011. A scrambling algorithm of image encryption based on Rubik's cube rotation and logistic sequence. In *Proceedings of the International Conference on Multimedia Signal Processing (CMSP'11)*, Vol. 1. IEEE, 312–315.

[45] Jiantao Zhou, Xianming Liu, Oscar C. Au, and Yuan Yan Tang. 2014. Designing an efficient image encryption-then-compression system via prediction error clustering and random permutation. *IEEE Transactions on Information Forensics and Security* 9, 1, 39–50.

Received April 2017; revised September 2017; accepted November 2017