Correspondence

A Virtual Image Cryptosystem
Based upon Vector Quantization

Tung-Shou Chen, Chin-Chen Chang, and Min-Shiang Hwang

Abstract—We propose a new image cryptosystem to protect image data. It encrypts the original image into another virtual image. Since both original and virtual images are signiﬁcant, our new cryptosystem can confuse illegal users. Besides the camouﬂage, this new cryptosystem has three other beneﬁts. First, our cryptosystem is secure even if the illegal users know that our virtual image is a camouﬂage. Second, this cryptosystem can compress image data. Finally, our method is more efﬁcient than a method that encrypts the entire image directly.

Index Terms—DES, image security, vector quantization, virtual image cryptosystem, steganography.

I. INTRODUCTION

Images are widely used in our daily lives. However, the more extensively we use images, the more important their security becomes. Thus, image security has become an important issue in the current computer world.

A cryptosystem is a useful tool for information security [3]. However, most traditional cryptosystems were only designed to protect text data. They are not suitable to encrypt images directly. This is because the image size is always much greater than that of text. So, traditional cryptosystems need much time to encrypt the image data. In addtion, for textual data, the decrypted result must be identical to the original text (i.e., plaintext). This requirement is, however, not always necessary for image data. Consider an image. Suppose it is only to be observed by human eyes. Then, a decrypted image that contains little distortion is still acceptable, since human eyes are not so sensitive.

Because of these differences between image and text, recently there have been several cryptosystems proposed for image security [1], [2], [6], [7]. These cryptosystems always encrypt the original image (i.e., plaintext image) into an encoded cipherimage. This encoded result protects our plainimage, but also piques the interest of illegal users. This is because the encoded image is always messy and meaningless. When illegal users pilfer the encoded image, they will see that this image is a fake.

In this cryptosystem, steganography.

Next, we randomly generate two vectors and . We call them the transformed-origin and the project-direction, respectively. In addition, we deﬁne a new parameter called the number of modiﬁed-bits as

\[ r = \left\lfloor \frac{2u \times b_e + n_v \times \log_2 n_v}{m_v \times (m_v - 1)} \right\rfloor \]

where is the number of bits needed to express the component value of , . , and . is the image size of . The numerator of is the number of bits that will be used to hide the encrypted data in our encryption phase that need to be hidden in the last pixels of . All , , and will be used in the following steps.

After the generation of , , and , we create the codebook of VQ based on these results and . We assign the value to the last l bits of each , and then generate a new set of vectors . Here, denotes the jth component value of . Since the last l bits of each will be used to hide the encrypted data in our encryption phase, these bits need to be uniﬁed in both encryption and decryption. Next, we absolutize the difference between each and , and project this result onto . The projected result is notably a one-dimensional (1-D) value. This transformation is formalized as

\[ T(V'_{i,j}) = \sum_{j=1}^{n} |V'_{i,j} - G_j| \times D_j. \]

After this transformation, we sort by their transformed values. Let the sorted result be , where for each .
are secure, too.

So, we embed these pixels of \( V'_1, V'_2, V'_3, \ldots, V'_{n_v} \) without them. Furthermore, their total size is much smaller than that of \( V'_1, V'_2, V'_3, \ldots, V'_{n_v} \). Thus we can encrypt them efficiently. In our cryptosystem, we select a DES-like method associated with the private key \( k \) for our encryption method. Suppose the DES-like method is secure. Then our \( w, h, n_v, G, \) and \( D \) are safe also. Let \( w^e, h^e, n^e_v, G^e \) and \( D^e \) denote the encrypted \( w, h, n_v, G, \) and \( D, \) respectively.

To protect the security of \( \{ I_1, I_2, I_3, \ldots, I_{n_o} \} \), we cannot encrypt them by the DES-like method directly, since their total size is still large. In our encryption phase, we encrypt them by a simple and quick operation “exclusive-OR,” which can be denoted by \( \oplus \). We first combine \( I_1, I_2, I_3, \ldots, I_{n_o} \) into a bit-string \( X \) and, moreover, generate another bit-string \( X_0 \) that contains \( G, D, G, D, \ldots \) only. Both the lengths of \( I \) and \( X_0 \) must be equal. Next, we define the encrypted indices \( I^e \) as \( I \oplus X \). Since XOR is a simple and basic operation, we can obtain \( I^e \) quickly. Furthermore, since \( G \) and \( D \) have been encrypted by the DES-like method, they are safe in our cryptosystem. Thus \( I_1, I_2, I_3, \ldots, I_{n_o} \) are secure, too.

Now, the last step of the encryption algorithm is to hide the above encrypted data into our virtual image. Since \( w, h, \) and \( n_v \) are the most basic parameters in our cryptosystem, most parameters are derived from them. Thus, \( w^e, h^e, \) and \( n^e_v \) must first be concealed such that they can be first decrypted by our decryption algorithm.

In our encryption algorithm, we embed them into the last bit of each component value in the first \( m \) pixels of \( V \). The bit-sizes of \( w^e, h^e, \) and \( n^e_v \) are newly defined by both the sender and the receiver. Next, we apply the same method to encrypt the other encrypted parameters \( G^e, D^e, \) and \( I^e \) into the residual \( m_v \times (m_v - 1) \) pixels of \( V \). However, the total bit-size of \( G^e, D^e, \) and \( I^e \) is \( 2m_v \times h^e + n_v \times \log_2 n_v \), which is usually greater than \( m_v \times (m_v - 1) \). So, we embed these \( 2m_v \times h^e + n_v \times \log_2 n_v \) bits into \( m_v \times (m_v - 1) \) pixels on average. That is, we hide \( G^e, D^e, \) and \( I^e \) into the last \( r \) bits of each component value in the last \( m_v \times (m_v - 1) \) pixels of \( V \). Here, \( r \) has been defined in (1). Last, the modified virtual image containing \( w^e, h^e, n^e_v, G^e, D^e, \) and \( I^e \) is then our ciphertext \( V^e \). The only difference between \( V \) and \( V^e \) is their last few bits of each pixel. Therefore, the distortion between \( V \) and \( V^e \) is limited, and \( V^e \) is always significant.

\( V^e \) and \( k \) are two important data to recover \( O \) after the encryption process. We thus need to send them to the receiver. Since \( V^e \) has been encrypted, it can be transmitted by the public channel. However, \( k \) is the secret key to decrypt \( V^e \). We must send it to the legal receiver by a secure channel [3].

**B. Decryption Algorithm**

The decryption algorithm is symmetric with our encryption one. First, we fetch \( w^e, h^e, \) and \( n^e_v \) from the last bit of each component value in the first \( m \) pixels of \( V^e \), and decrypt them by the DES-like method and \( k \). In this step, we obtain the original \( w, h, \) and \( n_v \). They are the most basic parameters in our cryptosystem. According to these values, we can evaluate the number of modified-bits \( r \) by (1). Here we assume that \( k \) is known by both the sender and receiver. Next, we fetch \( G^e, D^e, \) and \( I^e \) from the last \( r \) bits of each component value in the last \( m_v \times (m_v - 1) \) pixels of \( V^e \). Since \( G^e \) and \( D^e \) are also encrypted by the DES-like method, we need to employ this method to decrypt them, and obtain \( G \) and \( D \). \( I^e \) is, however, encrypted by \( G \) and \( D \), and an XOR operation. XOR is a symmetric operation. Thus we can decrypt \( I^e \) by \( G, D, \) and XOR, and acquire \( \{ I_1, I_2, I_3, \ldots, I_{n_o} \} \). Now, all parameters of our cryptosystem and the index set have been decrypted.

For generating our codebook, we first separate \( V^e \) into a set of \( v \)-dimensional vectors \( \{ V'_1, V'_2, V'_3, \ldots, V'_n_v \} \) based on \( w \) and \( h \). Next, we set the last \( r \) bits of each \( V'_j \) to be \( 2^{r - 1} \) since these bits have been changed in our encryption algorithm, and achieve \( \{ V''_1, V''_2, V''_3, \ldots, V''_{n_v} \} \). According to \( \{ V''_1, V''_2, V''_3, \ldots, V''_{n_v} \} \) and (2), we can acquire our codebook \( \{ V'_1, V'_2, V'_3, \ldots, V'_{n_v} \} \). Note that this codebook is identical with that in our encryption algorithm.

Finally, we conduct the VQ decoder to generate the decrypted image \( O^d \) based on the indices set \( \{ I_1, I_2, I_3, \ldots, I_{n_o} \} \) and the codebook \( \{ V'_1, V'_2, V'_3, \ldots, V'_{n_v} \} \). \( O \) and \( O^d \) are different, since VQ is a lossy data compression technology. However \( O \) and \( O^d \) must be close.

**III. EMPIRICAL TESTS AND SECURITY ANALYSIS**

**A. Empirical Tests**

To prove the feasibility of our virtual image cryptosystem, we conducted two experiments, which are described in this subsection. In these experiments, we employed “airplane” to be our original image \( O \). Its image size was always \( 512 \times 512 \) pixels, and we show it in Fig. 1. To encrypt this original image, we employed two virtual images \( V^1 \) and \( V^2 \) in our two experiments, respectively. Both of these images were images of Lena, but their image sizes were different. The size of the first virtual image \( V^1 \) was \( 256 \times 256 \) pixels (see Fig. 2), and that of \( V^2 \) was \( 360 \times 360 \) pixels. In our experiments, \( w, h, \) and \( v \) were defined to be \( 4, 4, \) and \( 16, \) respectively. That is, all images in our experiments were divided into \( 4 \times 4 \) pixel blocks. The number of
blocks in $O$ (i.e., $n_v$) was therefore $(512 \times 512)/(4 \times 4) = 16384$, and $n_v$ thus needed 16 b. Except for $n_v$, we assumed that all numbers were expressed by 8-b variables in our experiments, and $b_v$ was also set to be eight.

We evaluated the quality of the processed images by their peak signal-to-noise ratio (PSNR) [5] in our experiments. The larger the PSNR, the better the image quality will be. In general, a processed image is acceptable to human eyes if its PSNR is greater than 30 dB. Besides PSNR, we estimated the compression ratio of a processed image by the result of the compressed image size divided by the original size. The smaller the compression ratio was, the better was the compression effect that was achieved.

We applied $V^{-1}$ to be the virtual image of $O$ in our first experiment. In the virtual image cryptosystem, only $w$, $h$, $n_v$, $G$, and $D$ need be encrypted. Their total bit-size was $8 + 8 + 16 + 16 \times 8 + 16 \times 8 = 288$. Here the bit-sizes of both $G$ and $D$ were $16 \times 8$ since $G$ and $D$ were 16-dimensional vectors and each element in $G$ and $D$ was an 8-b variable. Since the number of processed bits per DES execution was 64, we only needed to execute DES $5(= [288/64])$ times to encrypt them. This result is much smaller than that needed to encrypt the entire original image. After the process of our encryption, the cipherimage of this experiment is presented in Fig. 3. Its PSNR is 37.87 dB. This value is much larger than our acceptable criterion of 30 dB. As for the compression ratio of this experiment, it equaled $(m_v \times m_v)/(m_o \times m_o) = (256 \times 256)/(512 \times 512) = 25\%$. Finally, after the decryption, the decrypted image is drawn in Fig. 4. Its PSNR is 30.22 dB. It is also greater than our criterion. So, this decrypted image can preserve the information of our original image.

In our second experiment, we conducted $V^2$ to be the virtual image of $O$. Note that the image size of $V^2$ is $360 \times 360$, which is greater than that of $V^{-1}$. In this case, the bit-sizes of $w$, $h$, $n_v$, $G$, and $D$ were still 8, 8, 16, 16 $\times$ 8, and $16 \times 8$, respectively. So, their total bit-size was not changed, and we also needed to execute DES five times to encrypt them. The cipherimage of this experiment is presented in Fig. 5. Its PSNR is 45.13 dB. This result is much better than that of our previous cipherimage. Besides, our decrypted image for this case is drawn in Fig. 6. Its PSNR is 31.36 dB. This is also better than that of our previously decrypted image. The compression ratio of this experiment was $(360 \times 360)/(512 \times 512) = 0.494 \pm 50\%$.

From the above experimental results, we obtain three benefits about the proposed cryptosystem. First, its cipherimage is always significant. So, it can confuse potential thieves. Second, the size of the cipherimage can be smaller than that of the original one. Thus, this cryptosystem can compress image data. Finally, this cryptosystem only needs a few DES-like processes to encrypt its important parameters. Thus, it is more efficient than to encrypt the entire original image directly. Also, we find that the image quality of our cryptosystem is inversely proportional to its compression rate. When the size of the virtual image is increased, the qualities of the cipherimages and the decrypted images will be improved, but the compression rate of this cryptosystem will be lost, and vice versa.

### B. Security Analyses

The camouflage is the first safeguard of our cryptosystem. In our cryptosystem, we employ a virtual image $V$ to camouflage our
original image. Since the distortion between $V$ and $V'$ is limited, $V'$ is also significant. Illegal users cannot detect that $V'$ is fake without any hint, even if they steal $V'$.

On the other hand, suppose the illegal users detect that $V'$ is fake, and want to break it. Then the outlaws may apply the following three types of attacks to break our cryptosystem. The first is the ciphertext-only attack [3]. In this attack, the illegal users are assumed to have only a cipherimage $V'$, and do not have the private key $k$. The outlaws cannot obtain $w, h, n, G$, and $D$ in this case since these parameters are encrypted by the DES-like method and $k$. Thus, the thieves cannot obtain our original image. However, suppose the illegal users try to guess $k$ by brute force. Let the bitsize of $k$ be 112. Then, $k$ has $2^{112}$ possible combinations. Here we assume that our cryptosystem employs double DES to encrypt $w, h, n, G$, and $D$. A private key has 56 bits in DES. So, the key $k$ has 112 bits in double DES. If the illegal users employ a 100 MIPS computer to conjecture $k$, the computational load is then $2^{112}/(100 \times 10^6 \times 60 \times 24 \times 365) = 1.646 \times 10^{18}$ years. This is, indeed, a very long time. Thus, our cryptosystem is secure for ciphertext-only attack.

The other two attacks are the known-plaintext and chosen-text attacks [3]. They are more religious than the ciphertext-only attack. In these two attacks, the illegal users are assumed to have obtained several plainimage and cipherimage pairs, and all of these pairs share a common key $k$.

In these cases, the outlaws can analyze these pairs to obtain the common key $k$, and correctly decrypt the next cipherimage if the sender still encrypts his next original image by $k$. To prevent these attacks, we define that our private key is disposable, i.e., it is a one-time pad system [3]. Since no common key exists in our cryptosystem, no one can break our cryptosystem by the known-plaintext or the chosen-text attack.

REFERENCES


A Deblocking Technique for Block-Transform Compressed Image Using Wavelet Transform Modulus Maxima

Tai-Chiu Hsung, Daniel Pak-Kong Lun, and Wan-Chi Siu

Abstract—In this work, we introduce a deblocking algorithm for Joint Photographic Experts Group (JPEG) decoded images using the wavelet transform modulus maxima (WTMM) representation. Under the WTMM representation, we can characterize the blocking effect of a JPEG decoded image as: 1) small modulus maxima at block boundaries over smooth regions; 2) noise or irregular structures near strong edges; and 3) corrupted edges across block boundaries. The WTMM representation not only provides characterization of the blocking effect, but also enables simple and local operations to reduce the adverse effect due to this problem. The proposed algorithm first performs a segmentation on a JPEG decoded image to identify the texture regions by noting that their WTMM have small variation in regularity. We do not process the modulus maxima of these regions, to avoid the image texture being “oversmoothed” by the algorithm. Then, the singularities in the remaining regions of the blocky image and the small modulus maxima at block boundaries are removed. We link up the corrupted edges, and regularize the phase of modulus maxima as well as the magnitude of strong edges. Finally, the image is reconstructed using the projection onto convex set (POCS) technique [2] on the processed WTMM of that JPEG decoded image. This simple algorithm improves the quality of a JPEG decoded image in the sense of signal-to-noise ratio (SNR) as well as visual quality. We also compare the performance of our algorithm to the previous approaches, such as CLS and POCS methods. The most remarkable advantage of the WTMM deblending algorithm is that we can directly process the edges and texture of an image using its WTMM representation.

Index Terms—Image enhancement, wavelet transforms.

I. INTRODUCTION

Transform coding is an efficient block-based image compression technique that has been widely used in the image compression industry. In particular, the discrete cosine transform (DCT) has been adopted as the basic compression algorithm of the Joint Photographic Experts Group (JPEG), Motion Picture Experts Group (MPEG), and others. For conventional transform coding, an image is first divided into a number of $n \times n$ nonoverlapped blocks. Each block is

Manuscript received June 6, 1996; revised August 12, 1997. This work was supported by Hong Kong Polytechnic University under Grant 340/808/A3/420. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Moncef Gabouj.

The authors are with the Department of Electronic Engineering, Hong Kong Polytechnic University, Hung Hom, Hong Kong (e-mail: enpcklu@polyu.edu.hk).

Publisher Item Identifier S 1057-7149/98$10.00 © 1998 IEEE