ON THE DESIGN OF AN EFFICIENT ENCRYPTION-THEN-COMPRESSION SYSTEM

Jiantao Zhou[1], Xianming Liu[2], and Oscar C. Au[3]

[1]Department of Computer and Information Science, University of Macau, Macau, China
[2]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
[3]Department of ECE, Hong Kong University of Science and Technology, Hong Kong, China

ABSTRACT

In many practical scenarios, image encryption has to be conducted prior to image compression. This has led to the problem of how to design a pair of encryption and compression algorithms such that compressing the encrypted image can still be efficiently performed. In this work, we propose a permutation-based image encryption method conducted over the prediction error domain. We also design an arithmetic coding (AC)-based approach to efficiently compress the encrypted image. It can be shown that the proposed scheme can provide reasonably high level of security. More notably, the compression performance on the encrypted image is only slightly degraded, compared with that of compressing the original, un-encrypted one. In contrast, most of the existing approaches induce significant penalty on the compression performance.

Index Terms— image encryption, image compression

1. INTRODUCTION

Consider an application scenario in which a sender Alice wants to securely and efficiently transmit an image I to a recipient Bob, via an un-trusted channel provider Charlie. Conventionally, this could be done as follows. Alice first compresses I into B, and then encrypts B into I_e using an encryption function $E_K(\cdot)$, where K denotes the secret key. Namely, $I_e = E_K(B)$. The encrypted file I_e is then passed to Charlie, who simply forwards it to Bob. Upon receiving I_e, Bob sequentially performs decryption and decompression to get a reconstructed image $\hat{I}$.

However, in many practical situations, the above *compression-then-encryption* paradigm needs to be reversed. As the sender, Alice might only care about the security of the image, while not the bandwidth usage. Hence, Alice may only encrypt I (without compression), and directly pass the encrypted image to Charlie for the actual transmission. This is especially true when Alice uses a resource-deprived device, because compression potentially leads to additional computational burden and battery consumption. In contrast, as the channel provider, Charlie is always interested in reducing the size of the date transmitted through the channel. It is therefore much desired if the compression task can be delegated by Charlie. A big challenge within such *encryption-then-compression* framework is that compression has to be conducted in the encrypted domain, as Charlie does not know the secret key K.

At the first glance, it seems to be impossible for Charlie to compress the encrypted data, since no signal structure can be exploited to enable a compressor. Although counter-intuitive, Johnson *et. al* demonstrated that the stream cipher encrypted data is compressible through the use of coding with side information principles, without compromising either the compression efficiency or the information-theoretic security [1]. In addition to the theoretical results, [1] also proposed a practical algorithm to losslessly compress the encrypted *bi-level* images. By using LDPC codes in various bit-planes and exploiting the inter/intra correlation, [2] suggested several methods for lossless compression of encrypted grayscale/color images. Later, [3] designed a progressive approach to losslessly compress grayscale/color images. Furthermore, [4] presented some algorithms to perform blind compression of encrypted videos. It should be noted that, compared with the traditional image compression algorithms applied to the original, un-encrypted grayscale/color images, the above methods induce significant coding loss.

In this work, our primary focus is on the practical design of a pair of image encryption and compression schemes, in such a way that compressing the encrypted image is *almost* equally efficient as compressing the original, un-encrypted one. Meanwhile, reasonably high level of security needs to be ensured. To this end, we propose a permutation-based image encryption approach conducted over the prediction error domain. A context-adaptive AC can then be applied to efficiently compress the encrypted data. Thanks to the nearly i.i.d property of the prediction error sequence, very small amount of compression penalty ($< 0.2\%$) will be introduced. In addition, due to the high sensitivity of prediction error sequence against disturbances, it can be shown that reasonably high level of security could be achieved.

The rest of this paper is organized as follows. Section 2 gives the details of the proposed system. Section 3 presents the performance and security evaluation. Experimental results are provided in Section 4. We conclude in Section 5.

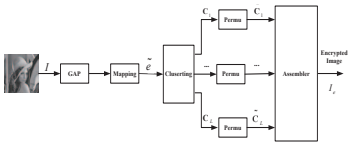

Fig. 1. Schematic diagram of image encryption

2. PROPOSED SYSTEM

In this section, we present the details of the three key components in our proposed encryption-then-compression system, namely, image encryption conducted by Alice, image compression conducted by Charlie, and the joint decryption and decompression conducted by Bob.

2.1. Image encryption via prediction error clustering and random permutation

From the system perspective, the design of the encryption algorithm should simultaneously consider the security and the ease of compressing the encrypted data. To this end, we propose an image encryption scheme operated over the prediction error domain. The schematic diagram of this encryption method is illustrated in Fig. 1. For each pixel $I_{i,j}$ of the image to be encrypted, a prediction $\bar{I}_{i,j}$ is first made by using an image predictor, e.g., GAP [5] or MED [6], according to its causal surroundings. In this work, we adopt the GAP as the image predictor, due to its excellent de-correlation capability. The prediction result $\bar{I}_{i,j}$ can be further refined to $\tilde{I}_{i,j}$ through a context-adaptive, feedback mechanism [5]. Consequently, the prediction error associated with $I_{i,j}$ can be computed by

$$e_{i,j} = I_{i,j} - \tilde{I}_{i,j} \tag{1}$$

Although for 8-bit images, the prediction error can potentially take any values in the range $[-255, 255]$, it can be mapped into the range $[0, 255]$, by considering the fact that the predicted value $\tilde{I}_{i,j}$ is available at the decoder side. From (1), we know that $e_{i,j}$ must fall into the interval $[-\tilde{I}_{i,j}, 255 - \tilde{I}_{i,j}]$, which only contains 256 distinct values. More specifically, if $\tilde{I}_{i,j} \leq 128$, we rearrange the possible prediction errors $-\tilde{I}_{i,j}, -\tilde{I}_{i,j}+1, \cdots, 0, 1, \cdots, \tilde{I}_{i,j}, \tilde{I}_{i,j}+1, \cdots, 255-\tilde{I}_{i,j}$ in the order $0, +1, -1, \cdots, +\tilde{I}_{i,j}, -\tilde{I}_{i,j}, \tilde{I}_{i,j}+1, \tilde{I}_{i,j}+2, \cdots, 255-\tilde{I}_{i,j}$. If $\tilde{I}_{i,j} > 128$, a similar mapping could be applied. Note that, in order to reverse this mapping, the predicted value $\tilde{I}_{i,j}$ is needed. Let us denote the mapped prediction error by $\tilde{e}_{i,j}$, which takes values in the range of $[0, 255]$.

As mentioned earlier, the proposed image encryption is performed over the domain of prediction error, i.e., $\tilde{e}_{i,j}$. Instead of treating all the prediction errors as a whole, we divide the prediction errors into L clusters based on a context-adaptive approach. To this end, we adopt the error energy estimator originally proposed in [5] as an indicator of the image local activities. More specifically, for each pixel location (i, j), the error energy estimator is defined by

$$\Delta_{i,j} = d_h + d_v + 2|e_{i-1,j}| \tag{2}$$

where

$$\begin{aligned} d_h &= |I_{i-1,j} - I_{i-2,j}| + |I_{i,j-1} - I_{i-1,j-1}| \\ &+ |I_{i,j-1} - I_{i+1,j-1}| \\ d_v &= |I_{i-1,j} - I_{i-1,j-1}| + |I_{i,j-1} - I_{i,j-2}| \\ &+ |I_{i+1,j-1} - I_{i+1,j-2}| \end{aligned} \tag{3}$$

The design of the cluster should simultaneously consider the security and better preparation for the subsequent compression. In an off-line training process, we collect a set of samples $(\tilde{e}, \Delta)$ from appropriate training images. A dynamic programming technique can then be employed to get an optimal cluster in minimum entropy sense, i.e., choose $0 = q_0 < q_1 < \cdots < q_L = \infty$ such that the following conditional entropy measure is minimized

$$-\sum_{0 \leq i \leq L-1} H(\tilde{e}|q_i \leq \Delta < q_{i+1}) p(q_i \leq \Delta < q_{i+1}) \tag{4}$$

where $H(\cdot)$ is the 1-D entropy function. The selection of the parameter L should balance the security and complexity. Larger L could potentially provide higher level of security because there are more possibilities for the attacker to figure out. However, it also incurs higher complexity of encryption. We heuristically find that $L = 16$ achieves a good balance between the above two conflicting factors. Note that the cluster configurations, i.e., all q_i, are publicly accessible. For each pixel location (i, j), the associated cluster index k can be determined by

$$k = \{k | q_k \leq \Delta_{i,j} < q_{k+1}\} \tag{5}$$

The procedure of performing the image encryption is as follows:

Step 1: Compute all the mapped prediction errors $\tilde{e}_{i,j}$.

Step 2: Divide all the prediction errors into L clusters $\mathbf{C}_k$, for $0 \leq k \leq L-1$, where k is determined by (5) and each $\mathbf{C}_k$ is formed by concatenating the mapped prediction errors in a raster-scan order.

Step 3: Reshape the prediction errors in each $\mathbf{C}_k$ into a 2-D block having four columns and $\lceil |\mathbf{C}_k|/4 \rceil$ rows, where $|\mathbf{C}_k|$ denotes the number of elements in $\mathbf{C}_k$.

Step 4: Perform two key-driven cyclical shift steps to each resulting prediction error block, and read out the data in raster-scan order to obtain the permuted cluster $\tilde{\mathbf{C}}_k$.

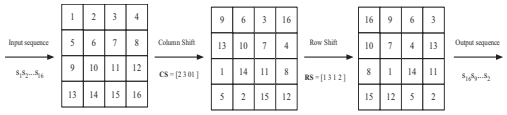

Fig. 2. An example of cyclical shifts.

Let $\mathbf{CS}_k$ and $\mathbf{RS}_k$ be the secret key vectors controlling the column and the row shift offsets for $\mathbf{C}_k$. Here, $\mathbf{CS}_k$ and $\mathbf{RS}_k$ are obtained from the key stream generated by a stream cipher, which means that the employed key vectors could be different, even for the same image at different encryption sessions. In Fig. 2, we show an example of the cyclical shift operations applied to a sequence $S = s_1 s_2 \cdots s_{16}$. The key vectors controlling the column and the row shift offsets are $\mathbf{CS} = [2\ 3\ 0\ 1]$ and $\mathbf{RS} = [1\ 3\ 1\ 2]$, respectively.

Step 5: Form the final encrypted image by concatenating all the permuted clusters $\tilde{\mathbf{C}}_k$, for $0 \leq k \leq L-1$, namely

$$I_e = \tilde{\mathbf{C}}_0 \tilde{\mathbf{C}}_1 \cdots \tilde{\mathbf{C}}_{L-1} \tag{6}$$

in which each prediction error is represented by 8 bits. As the number of prediction errors equals that of the pixels, the file size before and after the encryption is preserved.

Step 6: Pass I_e to Charlie, together with the length of each cluster $|\tilde{\mathbf{C}}_k|$, for $0 \leq k \leq L-2$.

This enables Charlie to divide I_e into L clusters appropriately. Compared with the file size of the encrypted data, the overhead induced by the length $|\tilde{\mathbf{C}}_k|$ is negligible.

2.2. Compression of encrypted image via adaptive AC

The compression of the encrypted file needs to be performed in a blind way, as Charlie does not have access to the secret key K. Assisted by the side information $|\tilde{\mathbf{C}}_k|$, for $0 \leq k \leq L-2$, Charlie can parse the received image I_e into L segments $\tilde{\mathbf{C}}_0, \tilde{\mathbf{C}}_1, \cdots, \tilde{\mathbf{C}}_{L-1}$. An adaptive AC can then be employed to losslessly encode each prediction error sequence $\tilde{\mathbf{C}}_k$, for $0 \leq k \leq L-1$, into a binary bit stream $\mathbf{B}_k$. Consequently, the final compressed and encrypted bit stream is produced by concatenating all $\mathbf{B}_k$, namely,

$$\mathbf{B} = \mathbf{B}_0 \mathbf{B}_1 \cdots \mathbf{B}_{L-1} \tag{7}$$

Similar to the encryption stage, the length of each $|\mathbf{B}_k|$ has to be sent to Bob as side information. The compressibility of each $\tilde{\mathbf{C}}_k$ relies on the fact that random permutation only changes the locations while not the values of prediction errors, leading to the preservation of the probability mass function (PMF) of prediction error sequence. The length of the resulting compressed bit stream can then be computed by

$$Len = |\mathbf{B}| + \sum_{k=0}^{L-2} \log_2 \lceil |\mathbf{B}_k| \rceil \tag{8}$$

where $|\mathbf{B}|$ and $|\mathbf{B}_k|$ denote the length (in bits) of $\mathbf{B}$ and $\mathbf{B}_k$, respectively.

2.3. Joint decryption and decompression

Upon receiving the compressed and encrypted bit stream $\mathbf{B}$, Bob aims to recover the original image I. According to the side information $|\mathbf{B}_k|$, Bob divides $\mathbf{B}$ into L segments $\mathbf{B}_k$, for $0 \leq k \leq L-1$, each of which corresponds to a cluster of prediction errors. For each $\mathbf{B}_k$, an adaptive arithmetic decoding can be applied to obtain the corresponding permuted prediction error sequence $\tilde{\mathbf{C}}_k$. A de-permutation operation is then employed to get back the original $\mathbf{C}_k$.

With all the $\mathbf{C}_k$, the decoding of the pixel values can be achieved in a raster-scan order. For each location (i, j), the associated error energy estimator $\Delta_{i,j}$ and the predicted value $\tilde{I}_{i,j}$ can be computed from the causal surroundings, which have already been decoded. Given $\Delta_{i,j}$, the corresponding cluster index k can be determined by using (5). The first unused prediction error in the kth cluster is selected as $\tilde{e}_{i,j}$, which will be used to get back $e_{i,j}$ according to $\tilde{I}_{i,j}$ and the mapping rule described in Section 2.1. Afterwards, a '*used*' flag will be attached to that prediction error. The reconstructed pixel value then becomes

$$\hat{I}_{i,j} = \tilde{I}_{i,j} + e_{i,j} \tag{9}$$

As the predicted value and the error energy estimator are both based on the causal surroundings, the decoder can exactly mimic the same operations of the encoder, ensuring perfect decoding without any error, i.e., $\hat{I}_{i,j} = I_{i,j}$.

3. SECURITY AND PERFORMANCE ANALYSIS

In this section, we present a brief analysis regarding the security and compression efficiency of our proposed system.

3.1. Security analysis

Recall that the key stream controlling the random permutation of $\mathbf{C}_k$ is generated by using a stream cipher. The only attack type applicable to our proposed system is therefore the ciphertext-only attack, in which the attacker can only access the ciphertext and attempts to recover the original image.

Since only the locations of prediction errors are randomized while their values remain intact, the attacker can still know the original histogram of prediction errors from the encrypted image. In other words, some statistical information about the image leaks. However, the number of distinct ways of permutation for each $\mathbf{C}_k$ is approximately

$$\frac{|\mathbf{C}_k|!}{\prod_{i=0}^{255} \left(\lfloor p(i) \cdot |\mathbf{C}_k| \rfloor ! \right)} \tag{10}$$

where $p(i)$ denotes the occurrence probability of value i in the cluster $\mathbf{C}_k$. In practice, the number given in (10) is very large, precluding practical brute-force attack. For instance, the number of distinct ways of permutation for the first cluster of Lena image is larger than 2^{256}. It should be noted that statistical information leakage is inevitable for any feasible encryption-then-compression systems. This is because an attacker can also perform blind compression on the encrypted data, and the size of the resulting compressed file has already been a statistical indicator of the original image. For instance, the Barbara image having more intensive statistical activities would lead to a larger file than Lena image.

Alternatively, the attacker may attempt to decode the encrypted file directly. For correct decoding of $I_{i,j}$, the attacker has to determine both the prediction error $\tilde{e}_{i,j}$ and the associated predicted value $\tilde{I}_{i,j}$. One way of guessing $\tilde{e}_{i,j}$ is to first estimate the cluster index k, and then select one element from $\tilde{\mathbf{C}}_k$. As both k and $\tilde{I}_{i,j}$ depend on the causal neighboring pixels that have been decoded, any previous decoding error may cause error propagation, influencing the correct decoding of consequent pixels. In this sense, the error propagation effect inherent in predictive coding helps improve security. Even if k can somehow be correctly determined, the attacker still needs to select one element from $\tilde{\mathbf{C}}_k$. As the distribution of the prediction errors is peaked at zero, the optimal estimation of $\tilde{e}_{i,j}$ in maximum likelihood (ML) sense is then zero. However, setting all $\tilde{e}_{i,j} = 0$ does not lead to a meaningful image.

Therefore, even though leakage of the statistical information occurs, the proposed permutation-based encryption is still practically useful in many scenarios where perfect secrecy is not required.

3.2. Compression performance

As a well-known fact, image predictors have strong de-correlation capability, making the prediction error sequence nearly i.i.d. In other words, only small amount of inter-dependence exists in the prediction error sequence. Compared with the traditional predictive coding in which the compression is conducted over the original, un-permuted prediction error sequence, the compression task of Charlie has to be performed over the *permuted* ones. It can be easily proved that a sequence with inter-dependence is more compressible than its i.i.d counterpart.

Clearly, permutation operations in the prediction error domain destroy the inter-dependence, making the resulted sequence $\tilde{\mathbf{C}}_k$ less compressible than its original, un-permuted counterpart. Fortunately, the inter-dependence left in each $\mathbf{C}_k$ is rather limited, thanks to the superior de-correlation capability of image predictors. Hence, the coding penalty caused by prediction error permutation is expected to be small. This will be verified experimentally in Section 4.

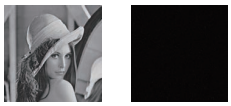

Fig. 3. Left: original; Right: encrypted.

4. EXPERIMENTAL RESULTS

In this section, the security and compression performance of our proposed system are evaluated experimentally. Fig. 3 shows the original Lena image and its encrypted version, from which we can see that the encryption method is effective in destroying the semantic meaning of the image.

In Table 1, the compression efficiency of our proposed method applied to the encrypted images is compared with the lossless rates given by CALIC, a benchmark of practically good lossless image codecs. Here 'B' and 'bpp' stands for bytes and bit per pixel, respectively. All the results of the proposed method are obtained by averaging over 10 random trials. It can be observed that very small amount ($< 0.2\%$) coding penalty is incurred. In contract, the coding loss of the method in [3] can be over 10% on average.

Table 1. Comparison of Lossless Compression Performance.

Image	Proposed	CALIC
Lena	134267 B (4.096 bpp)	134232 B (4.096 bpp)
Barbara	150369 B (4.589 bpp)	150223 B (4.584 bpp)
Man	142385 B (4.345 bpp)	142361 B (4.345 bpp)
Boat	134732 B (4.112 bpp)	134651 B (4.109 bpp)
Harbor	160567 B (4.900 bpp)	160487 B (4.898 bpp)
Airplane	121377 B (3.704 bpp)	121172 B (3.698 bpp)
Liver	81472 B (2.486 bpp)	81386 B (2.484 bpp)

5. CONCLUSIONS

In this paper, we have designed an efficient encryption-then-compression system. Within the proposed framework, the image encryption has been achieved via prediction error clustering and random permutation. Highly efficient compression of the encrypted data has been realized by an arithmetic coding approach. Both theoretical and experimental results have shown that reasonably high level of security has been obtained. More notably, our coding performance is very close to that of compressing the original, un-encrypted image.

6. ACKNOWLEDGEMENT

This work is supported by Start-up Research Grant (SRG) SRG023-FST13-ZJT of the University of Macau.

7. REFERENCES

[1] M. Johnson, P. Ishwar, V. M. Prabhakaran, D. Schonberg, and K. Ramchandran,"On compressing encrypted data," *IEEE Trans. on Signal Process.*, vol. 52, no. 10, pp. 2992-3006, Oct. 2004.

[2] R. Lazzeretti and M. Barni,"Lossless compression of encrypted grey-level and color images," in *Proc.16th Eur. Signal Processing Conf. (EUSIPCO 2008)*, Lausanne, Switzerland, Aug. 2008.

[3] W. Liu, W.J. Zeng, L. Dong, and Q.M. Yao,"Efficient compression of encrypted grayscale images," *IEEE Trans. on Imag. Process.*, vol. 19, no. 4, pp. 1097–1102, April 2010.

[4] D. Schonberg, S. C. Draper, C. Yeo, and K. Ramchandran,"Toward compression of encrypted images and video sequences," *IEEE Trans. on Inf. Forensics Security*, vol. 3, no. 4, pp. 749-762, Dec. 2008.

[5] X. Wu and N. Memon, "Context-based, adaptive, lossless image codec," *IEEE Trans. on Commun.*, vol. 45, pp. 437–444, 1997.

[6] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Trans. on Imag. Process.*, vol. 9, no. 8, pp. 1309–1324, 2000.