Localization and Correction of Corrupted Pixel Blocks in Noisy Encrypted Images

Pauline Puteaux, William Puech

To cite this version:

HAL Id: hal-03161514
https://hal.archives-ouvertes.fr/hal-03161514
Submitted on 6 Mar 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Localization and Correction of Corrupted Pixel Blocks in Noisy Encrypted Images

Pauline Puteaux and William Puech
LIRMM – Univ. Montpellier / CNRS
Montpellier, France

Abstract—Digital data such as images must be secured during transmission or cloud storage. Image encryption algorithms can be a solution to this problem, but these approaches are very noise sensitive. Because of the introduction of noise, the original image cannot be recovered, even if we know the secret key. In this paper, we propose a new noisy encrypted image correction algorithm containing a convolutional neural network (CNN) training stage and then, two main steps. After a direct decryption of a noisy encrypted image, the first step is to identify and localize the blocks that are probably incorrectly decrypted using a fine-tuned CNN. The second step of our proposed approach is to analyze and correct these blocks. Experimental results show that the proposed method can be used to blindly correct noisy encrypted images, while preserving the image structure and without increasing the original data size with additional information, unlike error correcting codes.

Index Terms—Multimedia security, image encryption, image denoising, deep learning, convolutional neural network, signal processing in the encrypted domain.

I. INTRODUCTION

During transmission or storage of encrypted images, it is often necessary to analyze or process them, without knowing the original content or the secret key used during the encryption phase [1]. When an encrypted image is corrupted during its transmission, even if the secret key is known, it becomes difficult to reconstruct the original image without errors.

Most of the previous methods propose to remove noise in noisy encrypted images by using error correcting codes (ECC) [2], [3]. ECC-based approaches consist of introducing redundancy in digital data. Check bits, computed from data using some algorithms, are added to the original bitstream. At the recipient side, check bits are derived from received data and compared with the received ones. If they are the same, no error is found, however, if an error is detected, an error correcting method needs to be applied. Error correction can be carried out using automatic repeat request (ARQ) or forward error correction (FEC). The ARQ technique consists of repeating the request for retransmission of corrupted data until the whole data is verified. The FEC approach is based on encoding data using error correcting code before transmission.

Furthermore, some papers have focused on error correction during the AES encryption algorithm, by parity determination and modification during the input and output of each round [4], [5]. Privacy-preserving error correction schemes have also been proposed. Hu et al. suggested using a double cipher to perform non-local means denoising [6]. SaghaisanNajadEsfahani et al. proposed to resort to secret sharing for wavelet denoising [7]. Recently, Pedrouzo-Ulloa et al. presented an error correction scheme based on 2-ring learning with errors, for homomorphically denoising images in the encrypted domain [8].

Other methods consist of performing statistical analysis of each block on the encrypted image during the decryption process to determine if it is correctly decrypted or not. Islam et al. explained how to correct noisy AES-encrypted images by calculating three statistical measurements: global variance method, mean local variance method and the sum of the squared derivative method [9]. Puteaux and Puech described an approach based on the local Shannon entropy measurement suitable for pixel blocks of very small sizes [10].

Moreover, note that most of the methods do not allow us to localize corrupted parts in the noisy encrypted image and do not preserve the original image structure or size.

In this paper, we propose a new method for correcting noisy encrypted using a block cipher images in order to first identify and localize, and then analyze and correct corrupted pixel blocks. Deep learning, in particular convolutional neural network (CNN), is involved in our algorithm. Indeed, in the last few years, deep learning architectures have obtained a significant performance gain with respect to conventional methods in many fields, such as multimedia security or clear image denoising [11]. In our application case, we show that it is actually possible to fine-tune a CNN to infer if a pixel block is clear or not. With such classification, a probable incorrectly decrypted block can be discriminated from a clear block of the original image. Thus, using this model, we can first identify and localize all probable incorrectly decrypted blocks after the direct decryption of a noisy encrypted image and then, we can perform the analysis and the correction of these blocks.

The rest of this paper is organized as follows. Section I describes our proposed algorithm of noisy encrypted image correction based on two main steps. Section III presents experimental results and discussion and finally, this paper is concluded in Section IV.

II. PROPOSED METHOD OF NOISY ENCRYPTED IMAGE CORRECTION

Starting from an encrypted image using AES algorithm in ECB mode, with blocks of $4 \times 4$ pixels, the encrypted image is corrupted during transmission due to channel noise. Therefore, during decryption, even with the key, it is not possible to correctly decrypt the noisy encrypted image directly. Indeed, all of the encrypted blocks that are noisy cannot be decrypted at all, this is because at least one bit has been flipped. In fact, it is
very difficult to discriminate a correctly decrypted block from an incorrectly decrypted one and, in particular, when the block size is very small. In this paper, we propose a new method to solve the problem of noisy image encryption decryption. Our algorithm is based in two main steps where deep learning, in particular CNN, is involved. We first identify and localize all the probable incorrectly decrypted pixel blocks, and then we analyze and correct them. Indeed, from a fine-tuned, pre-trained CNN, it is possible to learn if a block is clear or not. As illustrated in Fig. 1, in order to specialize a CNN, each image of a database of clear and encrypted (i.e. not clear) images is used and split into blocks of 4 × 4 pixels to train and fine-tune the CNN. Then we have a fine-tuned CNN (FT-CNN), which is able to classify a block as being clear, with a probability \( p_{\text{clear}}(\cdot) \), or not clear, with a probability \( 1 - p_{\text{clear}}(\cdot) \).

For the reconstruction of a noisy encrypted image, as illustrated in Fig. 1, from the directly decrypted image, based on the FT-CNN, the identification and localization step consists of separating the correctly decrypted blocks \( (p_{\text{clear}}(\cdot) = 1) \) from the probably incorrectly decrypted blocks \( (p_{\text{clear}}(\cdot) \neq 1) \).

**Algorithm 1:** Identification and localization of blocks that are probably incorrectly decrypted.

**Data:** Noisy encrypted blocks \( B \), classifier FT-CNN.

**Result:** List \( L \) of blocks \( B \) of the noisy encrypted image which have to be analyzed and corrected.

\[
L \leftarrow \emptyset; \quad \text{foreach} \ B \ \text{do} \quad \begin{cases} 
B_{\text{dec}} \leftarrow D_{\text{AES}}(B); \\
p_{\text{clear}}(B_{\text{dec}}) \leftarrow p_{\text{clear\_compute}}(B_{\text{dec}}; \text{FT-CNN}); \\
\text{if } p_{\text{clear}}(B_{\text{dec}}) \neq 1 \text{ then } \ L.\text{append}(B); 
\end{cases} \\
\text{return } L;
\]

Algorithm 1 presents with more details, the identification and localization step in order to obtain the list \( L \) of probably incorrectly decrypted blocks \( B \). Unlike correctly decrypted blocks, which can be directly used for the reconstruction of the original image, the probably incorrectly decrypted blocks must be analyzed so they can be corrected. Note that some blocks in the list \( L \) have a probability \( p_{\text{clear}}(\cdot) = 0 \), which means that we can assume that they are really incorrectly decrypted. In this case, we are pretty sure that, in these blocks, at least one bit has been flipped. But other blocks have a probability \( 0 < p_{\text{clear}}(\cdot) < 1 \), which means that these other blocks are maybe in clear, but with a higher or lower probability and therefore we must analyze and correct them too. For each probable noisy encrypted block of the list \( L \), since there is no information about the location and the number of corrupted bits, we have to test all the possible combinations. Without any assumptions, computational complexity is too high and performing a brute-force attack is not feasible, even for a block \( B \) of size \( 4 \times 4 \) pixels. This is why we propose to make two assumptions about the noise: the bit error rate due to noise is quite low; the noise is a white noise which uniformly alters the encrypted image. Under these two hypotheses, we can assume that one bit at most is flipped in each block of \( 4 \times 4 \) pixels of the encrypted image, which significantly reduces the complexity of the problem. Indeed, there are \( 1 + 128 \) possible configurations for each block, corresponding to the configuration of the original probable noisy encrypted block, plus the 128 configurations obtained by flipping one bit \( b_i \), \( (0 \leq i < 128) \). Algorithm 2 describes the analysis and correction process for each block. For each possible configuration of the encrypted block, the block is decrypted and used as input for the classifier FT-CNN, which returns a probability \( p_{\text{clear}}(\cdot) \). Among all the possible configurations, the one that obtains the highest probability to be in clear \( p_{\text{clear}}(\cdot)_{\text{max}} \) is considered as the expected block of the original image. This configuration is added to the reconstructed blocks of the original image as illustrated in Fig. 1.

**III. EXPERIMENTAL RESULTS AND DISCUSSION**

In order to fine-tune our CNN to decide if a block is clear or not, from the BOWS-2 database \([12]\) (grey-level images with a size of \( 512 \times 512 \) pixels) we randomly picked 100 images that remain in clear and 100 images which have been encrypted with AES-256 algorithm in ECB mode by blocks of \( 4 \times 4 \) pixels by using a different secret key for each image. We then obtain 1,638,400 blocks in clear and 1,638,400 encrypted blocks for training.
We used the Inception v3 model pre-trained on the ImageNet database, which achieves effective results in natural image classification tasks [13] and which is capable of extracting natural image features. Therefore, the previously computed weights of the CNN are kept fixed for our experiments and only the fully connected part is fine-tuned, substituting the last layer by a new randomly initialized softmax layer with two classes Clear and Not clear. Moreover, we upsample our images (4 x 4 pixels) to fit the required size of the Inception v3 model (299 x 299 pixels) by using nearest-neighbor interpolation in order to preserve block structure. Furthermore, as Inception v3 is pre-trained on the ImageNet database, the extracted features by the CNN are linked to intrinsic properties of natural images, which are not included in encrypted images. Indeed, the AES algorithm introduces confusion and diffusion in encrypted blocks and, as a consequence, there is no correlation between neighboring pixels and no patterns can appear. Therefore, we assume that our model can easily learn to discriminate the two classes Clear using probability $p_{clear}(\cdot)$ and Not clear using probability $1 - p_{clear}(\cdot)$.

In order to validate the learning capacity of the model, we use a cross-validation. Our whole block database is divided into four complementary batches of 819,200 images (409,600 clear blocks and 409,600 encrypted blocks). Three of these four batches are used for the training step and the obtained results are validated on the remaining batch. The confusion matrix presented in Table I, which is an average of the prediction results, shows that the prediction works very well since more than 99.7% of the total number of blocks are correctly predicted. After normalization, the very low variance values, $1.17 \times 10^{-9}$ for the class Clear and $2.70 \times 10^{-7}$ for the class Not clear, mean that our analysis is reproducible and there is no over fitting.

From the original image Lena ($512 \times 512$ pixels encoded with 256 grey-levels) illustrated in Fig. 2.a, we apply an AES encryption in ECB mode with a block of $4 \times 4$ pixels to obtain the encrypted image illustrated in Fig. 2.b. Note that there is no information about the original image content (PSNR of 8.56 dB). Fig. 2.c presents a noisy encrypted image of Fig. 2.b achieved using noise with a BER = $2.6 \times 10^{-3}$ which means that on average one bit every three blocks is randomly flipped. As framed in red, this represents 2,691, i.e. 16.42% of the total number of blocks in the image. Note that this BER value is relatively high in comparison with real life values. Indeed, depending on the transmission type, BER are between $10^{-12}$ for optical fiber and $10^{-4}$ for wireless transmission. PSNR between the original image and the noisy encrypted image remains low ($8.55$ dB) and PSNR of $33.91$ dB between the encrypted image and the noisy encrypted image indicates that the noise power is not negligible. Fig. 2.d illustrates that direct decryption without any analysis is not possible (PSNR of 16.33 dB), even if the secret key is known. This is due to the large number of noisy encrypted blocks which are then incorrectly decrypted. Moreover, without analysis it is not possible to localize incorrectly decrypted blocks and to discriminate them from correctly decrypted blocks of the original image.

Fig. 3 presents the results obtained using our proposed method in two steps: identification and localization, and then analysis and correction of probable incorrectly decrypted blocks. As illustrated in Fig. 3.a, the identification step of our proposed method consists of identifying and localizing the probable incorrectly decrypted blocks. The directly decrypted

### Algorithm 2: Analysis and correction of a block that is probably incorrectly decrypted.

**Data:** A block $B$ of the list $L$, classifier (FT-CNN).

**Result:** Reconstructed block $B_{clear}$.

1. $B_{clear} \leftarrow D_{AES}(B)$;
2. $p_{clear}(B_{clear})_{\text{max}} \leftarrow p_{clear\_compute}(B_{clear}, \text{FT-CNN})$;
3. for $i = 0 \text{ to } 127$ do
   a. $B_{dec} \leftarrow D_{AES}(B_{dec})$;
   b. $p_{clear}(B_{dec}) \leftarrow p_{clear\_compute}(B_{dec}, \text{FT-CNN})$;
   c. if $p_{clear}(B_{dec}) > p_{clear}(B_{clear})_{\text{max}}$ then
      - $p_{clear}(B_{clear})_{\text{max}} \leftarrow p_{clear}(B_{dec})$;
      - $B_{clear} \leftarrow B_{dec}$;
4. return $B_{clear}$.

### TABLE I

<table>
<thead>
<tr>
<th>Truth</th>
<th>Predicted</th>
<th>Clear</th>
<th>Not clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>408,562</td>
<td>1.038</td>
<td></td>
</tr>
<tr>
<td>Not clear</td>
<td>213</td>
<td>409,387</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 2.** Illustration of the problem of noisy encrypted image decryption: a) Original Lena image ($512 \times 512$ pixels, 256 grey-levels), b) Encrypted image, using the AES algorithm in ECB mode with blocks of $4 \times 4$ pixels, c) Noisy encrypted image, BER = $2.6 \times 10^{-3}$ (PSNR with the original image = 8.55 dB), d) Directly decrypted image (PSNR with the original image = 16.33 dB, 83.58% of correctly reconstructed blocks).
Fig. 3. Example of our method in two steps: a) Identification and localization of all probably incorrectly decrypted blocks from the directly decrypted image in Fig. 2.d, b) Analysis and correction of the noisy encrypted image in Fig. 2.c using the proposed method (PSNR with the original image = 45.66 dB, 99.98% of correctly reconstructed blocks).

The image (Fig. 2.d) is split into 16,384 blocks of 4 × 4 pixels. All the blocks which have a probability $p_{\text{clear}} = 1$ are considered as clear blocks of the original image. Such blocks represent 82.68% of the total number of blocks and are shown in clear in Fig. 3.a. We assume that all the other blocks are probably incorrectly decrypted due to the noise in the encrypted image. In fact 15.71% of these blocks have a probability $p_{\text{clear}} = 0$. This means that we can assume that these blocks are really incorrectly decrypted and have to be corrected (in yellow in Fig. 3.a). 1.61% of these blocks have a probability $0 < p_{\text{clear}} < 1$, which means that we cannot directly decide if they have been correctly decrypted or not. Therefore, we also have to consider them as probably incorrectly decrypted (in red in Fig. 3.a). To perform the correction of these two kinds of blocks, from the noisy encrypted image in Fig. 2.c, we generate, for each block, the 128 possible configurations by flipping, one by one, a bit of the block and decrypting it. For blocks with a probability $0 < p_{\text{clear}} < 1$, we also consider the initial configuration without modification. The associated probability to be clear for each configuration is computed using our classifier. The configuration that has the highest probability to be clear is then considered as the expected block of the original image represented in Fig. 2.a. After applying all the steps of our proposed method on the entire image, we are then able to reconstruct almost all the blocks of the original image (99.98% of the total number of blocks). As illustrated in Fig. 3.b, only three blocks remain incorrectly decrypted (0.02%) and the PSNR between the original image and the reconstructed image is high (45.66 dB).

Fig. 4 focuses on an area of the image where some errors remain after applying our proposed method. Blocks in blue are inferred as clear, and red frames highlight the location of the remaining incorrectly decrypted blocks after applying our proposed method. In Fig. 4.a, we can see that two of the incorrectly decrypted blocks in the reconstructed image (Fig. 3.b) are still inferred as encrypted. Indeed, their probability to be in clear is low ($p_{\text{clear}} < 0.5$). The third incorrectly decrypted block is identified as clear. Indeed, in a few cases, encrypted blocks are quite homogeneous and the prediction fails. Fig. 4.b illustrates the same area of Fig. 4.a, but using the original image (Fig. 2.a). In this case, the expected clear blocks which cannot be reconstructed have a very low probability to be in clear ($p_{\text{clear}} < 0.2$) and they are predicted as not clear. In fact, these blocks correspond to highly textured blocks of the original image and cannot be differentiated from incorrectly decrypted blocks.

Fig. 5 and Fig. 6 illustrate examples of clear and encrypted blocks respectively. In Fig. 5 top row, we can see that homogeneous blocks and blocks with a pattern are inferred as Clear, with $p_{\text{clear}} = 1$. At the end of the identification and localization step, they are considered as correctly decrypted blocks of the original image. Moreover, as shown in Fig. 6 top row, if the pixel distribution in the block is uniform and if pixel values are far from each others, blocks are inferred as Not clear, with $p_{\text{clear}} = 0$. In this case, we are pretty sure that these blocks are incorrectly decrypted and need to be corrected. Middle and top rows of Fig. 5 and Fig. 6 refer to blocks with $0 < p_{\text{clear}} < 1$. They are then considered as probably badly decrypted and taken as input of the analysis and correction step of our method. In Fig. 5 middle row, we present blocks in clear which are finally considered as expected blocks of the original image. This
means that $p_{\text{clear}}()$ is higher than the values associated to the 128 other possible block configurations. In Fig. 6 middle row, we represent encrypted blocks whose $p_{\text{clear}}()$ is smaller than the value associated to the expected block in clear. Blocks in the middle of the two figures are then finally correctly inferred during the analysis and correction step. In contrast, failed cases are presented in the bottom rows. Even for the human visual system, it is not easy to make the difference between clear blocks and encrypted blocks, especially in the case of highly textured clear blocks (Fig. 5 bottom row) and relatively clear blocks and encrypted blocks, especially in the case of 128 parameters, we can see that results are not as good as those obtained with our new proposed method. Indeed, seven blocks (0.04%) remain incorrectly decrypted, which means four more than with our proposed approach in this paper (PSNR = 41.84 dB). However, we can note that there are two incorrectly decrypted blocks with our proposed approach, Fig. 3.b, which are correctly decrypted using [10] as illustrated in Fig. 7.b. This means that a combination of these two approaches could improve the correction efficiency.

We also compared our proposed method with a standard error correcting approach based on Reed-Solomon (RS) codes [3], as presented in Table II. Reed-Solomon codes are denoted as RS$(n,k)$ with symbols of $m$ bits (for gray-level images, $m = 8$ bits). The RS encoder takes as input $k$ symbols of $m$ bits and adds parity symbols to build a codeword of $n$ symbols of $m$ bits. During the decoding step, up to $t = \frac{n-k}{2}$ erroneous symbols can be corrected, with no information on error location. For a gray-level image with a size of $512 \times 512$ pixels, in order to be able to correct 1 bit per block of $4 \times 4$ pixels, RS(255, 251) codes with symbols of 8 bits are used. Indeed, with such parameters, up to $t = 2$ bytes (16 bits) among 16 blocks of $4 \times 4$ pixels ($n = 251 \simeq 16 \times (4 \times 4)$) can be corrected. In Table II, by using RS codes, we can see that the format compliance property is not achieved and this would increase the data size by 1.56%. Conversely, our proposed error correction approach keeps the data in the intended format and preserves the original size, for the same amount of corrected bits per block.

Finally, the proposed approach has been applied on 100 images of $512 \times 512$ pixels, which means that 1,638,400 blocks of $4 \times 4$ pixels have been analyzed. On average, more than 99.95% of the blocks have been correctly reconstructed.
probable incorrectly decrypted blocks can be obtained in an automatic way, which is not possible with conventional methods using error correction codes (ECC). During the second step of our approach, the configurations associated to each probable incorrectly decrypted block are analyzed. Our fine-tuned CNN computes, for each configuration, the probability to be a clear block and the configuration which achieves the highest score is considered as the expected block of the original image. At the end of the process, almost all blocks are correctly decrypted and the reconstructed image is very similar to the original one, according to the obtained results in terms of PSNR. Moreover, our proposed approach is format-compliant and does not expand the original data size. With traditional ECC, if we would like to preserve the original size and to be format-compliant, then the quality of the final image will decrease.

In future work, we will try to use neighboring blocks in order to perform a better correction of the blocks with a low probability to be clear at the end of the process. Moreover, we are also interested in increasing the training database of the fine-tuned CNN before the second step of our method, using blocks which are identified as clear during the first step (light learning). Indeed, we suppose that adding the reconstructed blocks of the original image will help to improve the efficiency of the prediction.

### References


