Localization and Correction of Corrupted Pixel Blocks in Noisy Encrypted Images

Pauline Puteaux, William Puech

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]. SaghaianNejadEsfahani *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 4×4 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

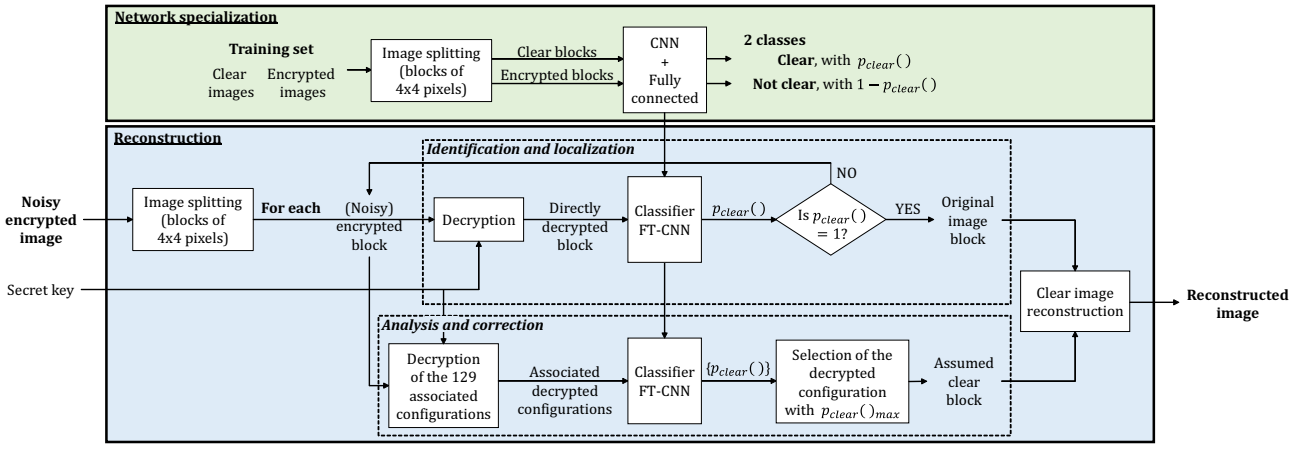


Fig. 1. Overview of the proposed method.

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 encrypted image 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 $p_{clear}()$, or not clear, with a probability $1 - p_{clear}()$.

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 ($p_{clear}() = 1$) from the probable incorrectly decrypted blocks ($p_{clear}() \neq 1$).

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 ← [ ];
foreach block B do
    B_dec ← D_AES(B);
    p_clear(B_dec) ← p_clear_compute(B_dec, FT-CNN);
    if p_clear(B_dec) ≠ 1 then
        L.append(B);
return L;
```

Algorithm 1 presents with more details, the identification and localization step in order to obtain the list L of probable incorrectly decrypted blocks B. Unlike correctly decrypted blocks, which can be directly used for the reconstruction of the original image, the probable incorrectly decrypted blocks must be analyzed so they can be corrected. Note that some blocks in the list L have a probability $p_{clear}() = 0$, 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 $0 < p_{clear}() < 1$, 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 4×4 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 4×4 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, (with $0 \leq i < 128$). 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 $p_{clear}()$. Among all the possible configurations, the one that obtains the highest probability to be in clear $p_{clear}()_{max}$ 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 512×512 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 4×4 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.

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}.
$B_{clear} \leftarrow D_{AES}(B)$;
$p_{clear}(B_{clear})_{max} \leftarrow$ *p_clear_compute*$(B_{clear},$ FT-CNN);
for $i = 0$ **to** 127 **do**
  $B_{dec_i} \leftarrow D_{AES}(B_{\bar{b}_i})$;
  $p_{clear}(B_{dec_i}) \leftarrow$ *p_clear_compute*$(B_{dec_i},$ FT-CNN);
  if $p_{clear}(B_{dec_i}) > p_{clear}(B_{clear})_{max}$ **then**
    $p_{clear}(B_{clear})_{max} \leftarrow p_{clear}(B_{dec_i})$;
    $B_{clear} \leftarrow B_{dec_i}$;
return B_{clear};

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 (4×4 pixels) to fit the required size of the Inception v3 model (299×299 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 $p_{clear}()$ and **Not clear** using probability $1-p_{clear}()$.

True \ Predicted	**Clear**	**Not clear**
Clear	$408,562$	$1,038$
Not clear	213	$409,387$

TABLE I
CONFUSION MATRIX, MEAN CLASSIFICATION FOR THE CROSS-VALIDATION (IN NUMBER OF IMAGES).

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 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, 1.17×10^{-9} for the class **Clear** and 2.70×10^{-7} for the class **Not clear**, mean that our analysis is reproducible and there is no over fitting.

From the original image *Lena* (512×512 pixels encoded with 256 grey-levels) illustrated in Fig. 2.a, we apply an AES encryption in ECB mode with a block of 4×4 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.

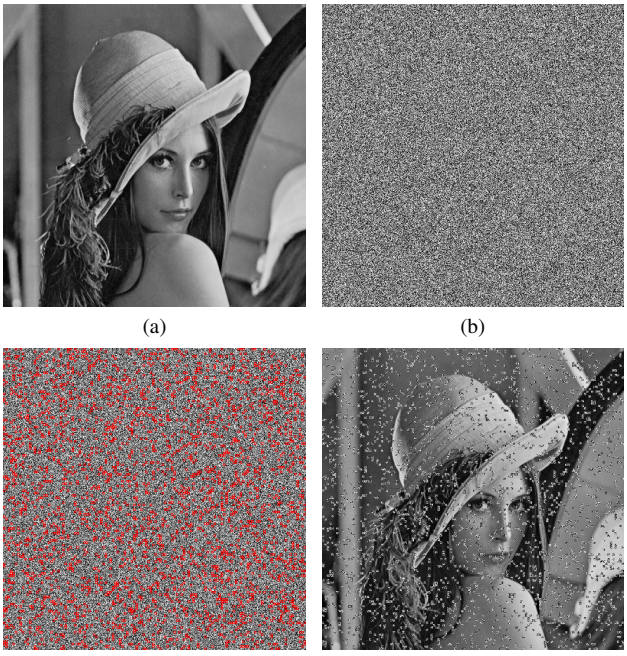
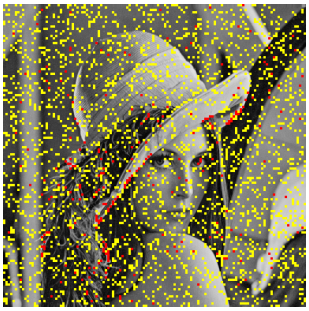


Fig. 2. Illustration of the problem of noisy encrypted image decryption: a) Original *Lena* image (512×512 pixels, 256 grey-levels), b) Encrypted image, using the AES algorithm in ECB mode with blocks of 4×4 pixels (PSNR with the original image $= 8.56$ dB), c) Noisy encrypted image, BER $= 2.6 \times 10^{-3}$ (PSNR with the original image $= 8.55$ dB, PSNR with the encrypted image $= 33.91$ dB), d) Directly decrypted image (PSNR with the original image $= 16.33$ dB, 83.58% of correctly reconstructed blocks).

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



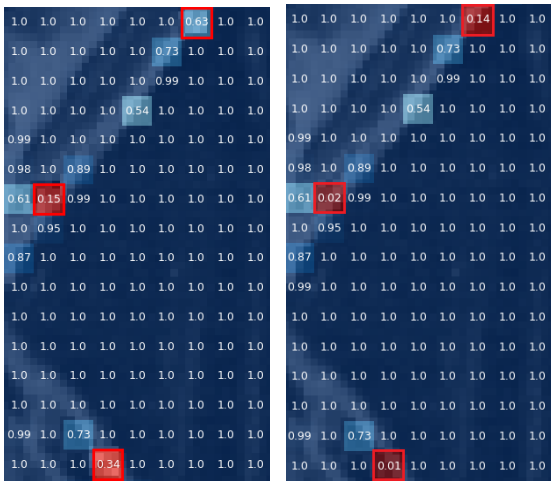
(a)

(b)

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).

image (Fig. 2.d) is split into $16,384$ blocks of 4×4 pixels. All the blocks which have a probability $p_{clear}() = 1$ 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 $p_{clear}() = 0$. 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 $0 < p_{clear}() < 1$, 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 $0 < p_{clear}() < 1$, 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 ($p_{clear}() < 0.5$). 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 ($p_{clear}() < 0.2$) 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.



(a) (b)

Fig. 4. Associated probability to be in clear for each block of a part of the: a) Reconstructed image using the proposed method (Fig. 3.b), b) Original image in clear (Fig. 2.a).

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 $p_{clear}() = 1$. 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 $p_{clear}() = 0$. 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 $0 < p_{clear}() < 1$. 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 $p_{clear}()$ is higher than the values associated to the 128 other possible block configurations. In Fig. 6 middle row, we represent encrypted blocks whose $p_{clear}()$ 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 homogeneous encrypted blocks or encrypted blocks with a pattern (Fig. 6 bottom row). Furthermore, note that we cannot use a threshold to discriminate clear and encrypted blocks, because some encrypted blocks have a higher probability to be clear ($p_{clear}()$ values) than some clear blocks.

(a) $p_{clear}() = 1$ (b) $p_{clear}() = 1$ (c) $p_{clear}() = 1$

(d) $p_{clear}() = 0.97$ (e) $p_{clear}() = 0.92$ (f) $p_{clear}() = 0.86$

(g) $p_{clear}() = 0.14$ (h) $p_{clear}() = 0.02$ (i) $p_{clear}() = 0.01$

Fig. 5. Examples of clear blocks: top row) Inferred as **Clear** after the identification and localization step, middle row) With $0 < p_{clear}() < 1$, and considered as expected blocks of the original image after the analysis and correction step, bottom row) With $0 < p_{clear}() < 1$, and considered as encrypted after the analysis and correction step.

In order to make a comparison with current state-of-the-art methods, Fig. 7 illustrates results obtained from approaches based on a local entropy analysis and which exploits the property that if a block is clear then it has a lower Shannon entropy value than an encrypted block. Fig. 7.a illustrates a reconstructed image using a method of correction based on the zero-order entropy without quantization. We can note that there are a lot of blocks which are incorrectly decrypted ($3,662$, *i.e.* 22.35%) and the PSNR value with the original image is low ($14.89\ dB$). Fig. 7.b. is obtained using the best parameters proposed in [10], which are distance map entropy and a quantization on 8 grey-levels. With these optimal parameters, we can see that results are not as good as those obtained with our new proposed method. Indeed, seven blocks (0.04%)

(a) $p_{clear}() = 0$ (b) $p_{clear}() = 0$ (c) $p_{clear}() = 0$

(d) $p_{clear}() = 0.04$ (e) $p_{clear}() = 0.15$ (f) $p_{clear}() = 0.21$

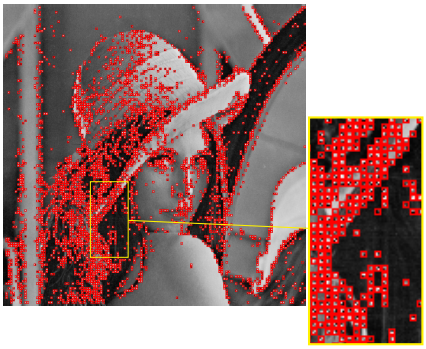
(g) $p_{clear}() = 0.84$ (h) $p_{clear}() = 0.72$ (i) $p_{clear}() = 0.55$

Fig. 6. Examples of encrypted blocks: top row) Inferred as **Not clear** after the identification and localization step, middle row) With $0 < p_{clear}() < 1$, and considered as encrypted after the analysis and correction step, bottom row) With $0 < p_{clear}() < 1$, and considered as the expected blocks of the original image after the analysis and correction step.

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 grey-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 grey-level image with a size of 512×512 pixels, in order to be able to correct 1 bit per block of 4×4 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 4×4 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 512×512 pixels, which means that $1,638,400$ blocks of 4×4 pixels have been analyzed. On average, more than 99.95% of the blocks have been correctly reconstructed.



(a)

(b)

Fig. 7. Correction of the noisy encrypted image in Fig. 2.c using local Shannon entropy measurement in blocks of 4×4 pixels: a) Reconstructed image using zero-order entropy and without quantization, *i.e.* by considering 256 grey-levels (PSNR with the original image $= 14.89\ dB$, 77.65% of correctly reconstructed blocks), b) Reconstructed image using distance map entropy and quantization, *i.e.* by considering 8 grey-levels [10] (PSNR with the original image $= 41.84\ dB$, 99.96% of correctly reconstructed blocks).

Approach	Original data size	Transmitted data size	#corrected bits/block	Format compliant
RS(255,251)	257 kB	261 kB	1	No
Proposed	257 kB	257 kB	1	Yes

TABLE II
COMPARISON BETWEEN USAGE OF RS(255,251) ERROR CORRECTING CODES AND PROPOSED METHOD OF NOISY ENCRYPTED IMAGE CORRECTION.

Note that even if some blocks are incorrectly reconstructed, image quality remains high compared to the original one, as indicated by a PSNR $> 40\ dB$).

IV. CONCLUSION

In this paper, we have shown that a fine-tuned CNN is efficient to infer if a block of pixels is clear or not. However, the simple use of a CNN is not sufficient to correct noisy encrypted images. For this purpose, in our proposed approach, we develop a new algorithm which is composed of two main steps: corrupted pixel block identification and localization first, then analysis and correction. In the first step of our method, directly from a noisy decrypted image, one can make the distinction between clear blocks and probable incorrectly decrypted ones. At the end of this step, a location map of 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

[1] Z. Erkin, A. Piva, S. Katzenbeisser, R. L. Lagendijk, J. Shokrollahi, G. Neven, and M. Barni, "Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing," *EURASIP Journal on Information Security*, vol. 2007, pp. 17, 2007.

[2] S. B. Wicker, *Error control systems for digital communication and storage*, vol. 1, Prentice hall Englewood Cliffs, 1995.

[3] R. H. Morelos-Zaragoza, *The art of error correcting coding*, John Wiley & Sons, 2006.

[4] K. Wu, R. Karri, G. Kuznetsov, and M. Goessel, "Low cost concurrent error detection for the advanced encryption standard," in *35th IEEE International Test Conference (ITC)*, 2004, pp. 1242–1248.

[5] V. Ocheretnij, G. Kouznetsov, M. Gossel, and R. Karri, "On-line error detection and bist for the AES encryption algorithm with different S-box implementations," in *11th IEEE International On-Line Testing Symposium (IOLTS)*, 2005, pp. 141–146.

[6] X. Hu, W. Zhang, H. Hu, and N. Yu, "Non-local denoising in encrypted images," in *1st International Conference on Internet of Vehicles (IOV)*. Springer, 2014, pp. 386–395.

[7] S. M. SaghaianNejadEsfahani, Y. Luo, and S.-C. S. Cheung, "Privacy protected image denoising with secret shares," in *19th IEEE International Conference on Image Processing (ICIP)*, 2012, pp. 253–256.

[8] A. Pedrouzo-Ulloa, J. R. Troncoso-Pastoriza, and F. Pérez-González, "Image denoising in the encrypted domain," in *8th IEEE International Workshop on Information Forensics and Security (WIFS)*, 2016, pp. 1–6.

[9] N. Islam, Z. Shahid, and W. Puech, "Denoising and error correction in noisy AES-encrypted images using statistical measures," *Signal Processing:Image Commun.*, vol. 41, no. C, pp. 15–27, Feb. 2016.

[10] P. Puteaux and W. Puech, "Noisy encrypted image correction based on Shannon entropy measurement in pixel blocks of very small size," in *26th European Signal Processing Conference (EUSIPCO)*, 2018.

[11] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 341–349.

[12] P. Bas and T. Furon, "Image database of BOWS-2," http://bows2.ec-lille.fr/.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.