B07902088
李彦廷

Implementing Reversible Data Hiding with MSB-Replacement Approach

Reversible data hiding, as the name suggests, refers to the process of hiding additional data into some encrypted medium, such as encrypted images for this attempted replication of one of the schemes proposed by P. Puteaux and W. Puech (2018) [1], utilizing the chaotic generator outlined in Shujun et al (2001) [2]. While some modifications and interpretations were made, implementations largely followed [1-2], yielding less than favorably comparable results than that is reported in [1] in some aspects.

This report is organized as thus. MSB-based reversible data hiding method proposed by [1] is first outlined, followed by the chaotic generator from [2] with some brief overviews on the piecewise linear chaotic map (PWLCM) used to realize [2] and the evaluation metrics of the resulting ciphertext images and reconstructed images, such as PSNR. Finally, the result of my attempt is presented.

MSB-Based Reversible Data Hiding

Reversible data hiding is usually, at the time of writing of [1], done with LSB-substitution of the encrypted bytes, wherein the bits of the secret message to embed are extracted and placed in the LSB of each byte. However [1] have mentioned that a MSB-substitution approach has its merits, such as in its easier handling when preprocessing the plaintext by means of pixel prediction, which is a necessary step to reconstruct the substituted bit. Given that embedding occurs without access to either the source image or the encryption key used to generate the ciphertext image, it makes balancing embedding capacity with reconstructed image quality difficult whilst maintaining the same file size - that is, no extra information is tacked on. Two schemes are presented, termed CPE-HCRDH and EPE-HCRDH; both shares a general overall approach, with differences originating from different specialization, the former with maximizing embedding capacity and the latter with maximizing reconstructed image quality. [1]

Method Overview

General pipeline consists of four parts: prediction error detection, image encryption, data encryption and embedding, and finally the reconstruction of source image and retrieval of hidden data - the two of which may be done independently or jointly.
With the aforementioned need for predicting displaced bits, errors in predicting the lost MSB of the original image need to be detected and handled with. Given a particular MSB-predictor function, the image is first searched for prediction errors, defined as when $|pred(i,j) - p(i,j)| \geq |pred(i,j) - inv(i,j)|$ where $p(i,j)$ is the pixel value, $pred(i,j)$ the predicted pixel value and $inv(i,j)$ the original pixel value with an inverted MSB at (i,j).
Image encryption utilizes a 1D PWLCM, whose parameters are given by the image key K_e, to generate pseudo-random bytes to encrypt the image data via XOR-operation. Data embedding is done by inserting the bits from the ciphertext data - encrypted with key K_w- into the MSB of the

bytes of the image, with exception that the first pixel, $p(0, 0)$, should be left untouched. Retrieval of message is done by extracting the MSBs from the image, then decrypted with K_w. The source image is obtained by first decrypting the remaining bits aside from MSB, then predicting the MSB using the predictor function used in earlier step, by computing the difference between possible pixel values and the predicted pixel value, then taking the result yielding the least absolute difference. [1]

CPE-HCRDH Approach

Only the prediction step differs from the general overview. Given a predictor averaging the left pixel and the top pixel relative to the current (i, j) position, the image is first preprocessed to eliminate all prediction errors by adjusting the pixels causing said errors. [1]

EPE-HCRDH Approach

Prediction errors are handled after encryption in this approach, thus causing subsequent embedding and reconstruction to differ from the general overview. After first marking down the location with prediction errors, image encryption is done. Consider the ciphertext in blocks of 8 pixels, should an error occur inside a block, said block will be bounded by blocks of all bytes whose MSB is 1. Then the embedding of data will use the remaining free blocks to embed their bits in the same manner as described in the overview procedure. Blocks of other sizes are applicable, though the authors have noted the potential of confusing message data and flagging data with smaller block sizes. Data and image recovery would follow the aforementioned scheme. [1]

Digital Chaotic Ciphers and 1D PWLCM

Image encryption in [1] used 1D PWLCM, with piecewise linear map (PWLM) defined as a "map composed of multiple linear segments, where limited breaking points are allowed". An example would be:

$$F(x) = \begin{cases} \dfrac{x}{p}, & x \in [0, p), \\ \dfrac{x-p}{0.5-p}, & x \in [p, 0.5], \\ F(1-x, p), & x \in [0.5, 1). \end{cases}$$ [3]

where $p \in (0, 0.5)$. [4] Specifically, the implementation I ended up with is using the framework from [2] with the above map for the digital chaotic cipher.

Digital chaotic ciphers generally have two methods of deployment: stream ciphers and block ciphers. Block cipher sources their control parameters from the input plaintext, then iterates a set m times, transforming the input to ciphertext. Stream cipher generates some sequence that is then used to perturb the plaintext; the encryption method proposed in [1] uses such an approach. For such stream ciphers, there are generally two major classes of architecture for sequence generation: using bits from the chaotic orbits, or custom decisions based on intervals the orbit falls into. [2]

There exists some limitations regarding digital chaotic ciphers in their implementation and design. Given that the systems need to iterate several times in order to ensure their security, coupled with floating point operations that may be expensive in their computation time, poorly designed ciphers would be too slow for real-time usage. As such, [2] suggests the use of fixed point arithmetic and the combined usage of multiple chaotic systems - such as PWLCM for their faster computation compared to other methods - as most reviewed in [2] were noted to be reliant on a single chaotic system, and due to the deterministic nature of these systems, may not be as resistant to orbit probing as combined systems have. [2]

One main limitation regarding digital realization of chaotic ciphers would be in its discretization of something otherwise defined on a continuous spectrum. As such, unfavorable behaviors may occur, such as shorter cycle length, skewed distribution, etc. [2] [3] Three approaches may be taken to help alleviate this issue of degradation: using more precision, using multiple chaotic systems, and using a perturbation approach; the perturbation approach is recommended over the other two choices. Perturbation-based approach aims to extend the cycle length of the system, which is especially of notable concern when the systems require multiple iterations before outputting a value, by introducing 'additional noise' at a particular interval Δ. This may be done by perturbing either the

- Chaotic orbit, $(x+1) = F_n(x(i)) \oplus S(i)$, or the
- Control parameters, $(x+1) = F_n(x(i) \oplus S(i))$, or

both together, where $S(i)$ is the perturbing signal generated by some pseudo random number generator (PRNG) with uniform distribution, which is applied (an example would be XOR-operation) every Δ iterations of the chaotic system. [3]

Chaotic PRNG

One of the applications of the 1D PWLCM is in creating a chaotic PRNG. Given the aforementioned discussion regarding degradation due to discretization, approaches such as the perturbation-based method should be incorporated. There are two schemes regarding such usages: directly using the iterated chaotic orbit after post processing it, or used to strengthen an existing PRNG. [3] In the former scheme, post-processing after obtaining a result from some m-iterated chaotic system may be any manipulation of the bits obtained. For a simple partial extraction of the bits, it is recommended to aim for the middle bits for two reasons. Firstly, the chaotic states are more dependent on bits with higher significance compared to that of bits with lower significance, hence it is advised to not extract higher bits. Secondly, degradation occurs mainly in the lower bits. As such, middle bits in the range $[\ \lfloor 2n/3 \rfloor , \lfloor n/3 \rfloor\ ]$ are advised. [3]

CCS-PRBG

The CCS-PRBG system proposed in [2] is a chaotic-orbit-perturbing approach using post processing for its output from the underlying chaotic systems. [2] It involves the use of two different 1D chaotic maps, $F_1(x_1, p_1)$ and $F_2(x_2, p_2)$, and let $x_1(i + 1) = F_1(x_1(i), p_1)$ and $x_2(i + 1) = F_2(x_2(i), p_2)$, then the bit is generated by the following function

$$g(x_1, x_2) = \begin{cases} 1, & x_1 > x_2 \\ \text{no outut}, & x_1 = x_2 \\ 0, & x_1 < x_2 \end{cases}$$

as $bit\ i = k(i) = g(x_1(i), x_2(i))$, with the maps $F_1(x_1, p_1)$ and $F_2(x_2, p_2)$ satisfying the requirements outlined in section 2.1 of [2]. The system works as follows. Pseudo random numbers are generated from both $F_1(x_1, p_1)$ and $F_2(x_2, p_2)$, wherein the l lowest bits of these results are XOR-ed by perturbing signals S_1 and S_2 in intervals of Δ_1 and Δ_2 generated by their own respective PRNGs, respectively, before being fed as inputs to the bit-generating function $g = \{k(i)\}$. It is advised to discard the first of such m bits of $\{k(i)\}$, wherein m is recommended to be $m > max(\Delta_1, \Delta_2)$. PRNGs may be implemented with linear congruential generators, and their perturbing length l should satisfy both: $l << n$ where n is the number of working precision, and that $l \geq \lceil \lambda \cdot log_2 e \rceil = \lceil 1.44\lambda \rceil$ where λ is the Lyapunov exponent of the chaotic map [2], which for 1D PWLCM with onto property - this condition of which is satisfied in the requirement of the maps utilized here - is proposed to be

$$\lambda = -\sum_{i=1}^{m} \|C_i\| \cdot ln\, \|C_i\|$$

where: $C_i = F_i(x)$ for $i = 1 \sim m$ and $\{C_i\}_{i=1}^{m}$ is a partition of X, the interval to which the map $F: X \rightarrow X$ is defined on. [3]

Given that this proposed system is centered around the use of different chaotic maps, should these two maps be equivalent in all parameters except their p value, then one may get $k'(i) = k(i)$ with $g(x_2, x_1) = g(x_1, x_2)$ as the function g relies on the comparison of the two underlying maps F_1 and F_2. Hence, PRNGs or perturbing intervals should be different for the two maps. Optionally, an additional perturbing of the output of g may be done with an additional PRNG. [2]

Implementation

The code itself is written in C++, with OpenCV for image processing, and the statistical analyses are done in Matlab.

CPE-HCRDH approach in [1] is implemented with the following deviations from the paper. Since [1] provided no implementation details, the 1D Chaotic generator used in image encryption uses the CCS-PRBG from [2], wherein a byte is obtained by concatenating the generated bits from MSB to LSB. The CCS-PRBG follows the method outlined in [2], with the following parameters:

- Message key, K_w = 0.947799
- Image key, $K_e(p, p^*, x_0, x_0^*, \Delta_1, \Delta_2)$
  = (0.252082, 0.172189, 0.112314, 0.185845, 7, 21)
  where (p, p^*) is the p-value, (x_0, x_0^*) the initial conditions and (Δ_1, Δ_2) the perturbing intervals for 1D PWLCM for F_1, F_2 respectively
- $m = max(\Delta_1, \Delta_2) + 1$
- $l = 10$, as I was simply unable to understand how to properly calculate the Lyapunov exponent of the 1D PWLCM used here, although its formulation is given in [3].

All the above values are generated using built-in libraries in C++. The PRNGs used in generating perturbing signals uses std::minstd_rand0 in C++, seeded by the p-value.

Furthermore, during the data embedding process, since the predictor uses neighbouring pixels to the left and top of itself to predict MSBs, the first row and first column of the image are not used in embedding. All calculations are realized with floating points, specifically, doubles.

Evaluation

The following is the result of conducting a CPE-HCRDH approach on grayscale Lena and Cameraman images of dimensions 512x512.

Image (512x512)	Horizontal Correlation	Vertical Correlation	NPCR (%)	UACI (%)	PSNR (dB)	SSIM
Lena, Original	0.9707	0.9842	x	x	x	x
Lena, Ciphertext	0.0147 (-0.0125)	0.0031714 (0.0224)	$3.8147\text{x}10^{-4}$ [99.62 / 99.62]	$2.2290\text{x}10^{-4}$ [15.02 / 15.02]	37.5238	0.9968
Cameraman , Original	0.9845	0.9893	x	x	x	x
Cameraman , Ciphertext	0.0025 (-0.0178)	-0.0017 (0.0073)	$3.8147\text{x}10^{-4}$ [99.62 / 99.62]	$2.0644\text{x}10^{-4}$ [17.39 / 17.39]	37.4536	0.9956

Table I. Replication Result on Lena and Cameraman images of dimension 512x512. Values in brackets of horizontal and vertical correlation refers to the result with ciphertext images after data embedding. Values in square brackets of NPCR and UACI refers to the result when each test is computed between the source image and said ciphertext image before / after data embedding.

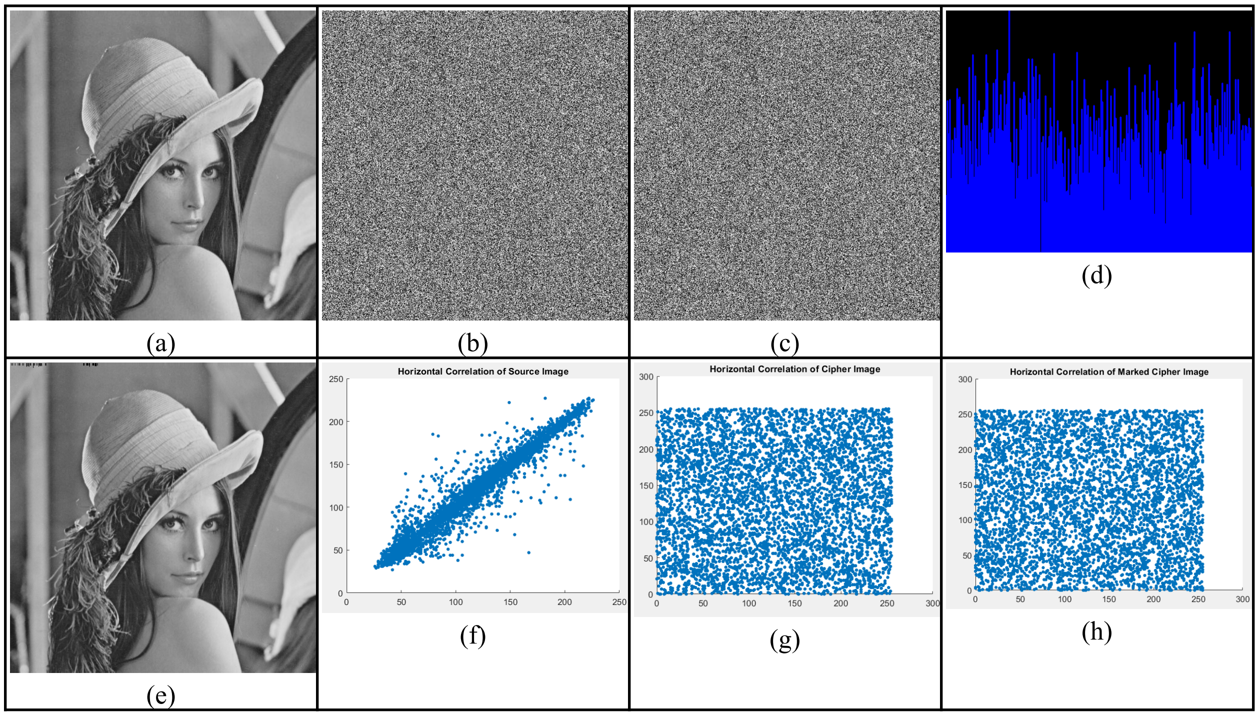

Fig 1. (a) source image; (b) ciphertext image before embedding; (c) ciphertext image after embedding; (d) histogram of the pixel value distribution in the ciphertext embedded with data; (e) reconstructed image; (f) horizontal correlation of source image; (g) horizontal correlation of ciphertext image; g) horizontal correlation of ciphertext image after embedding data.

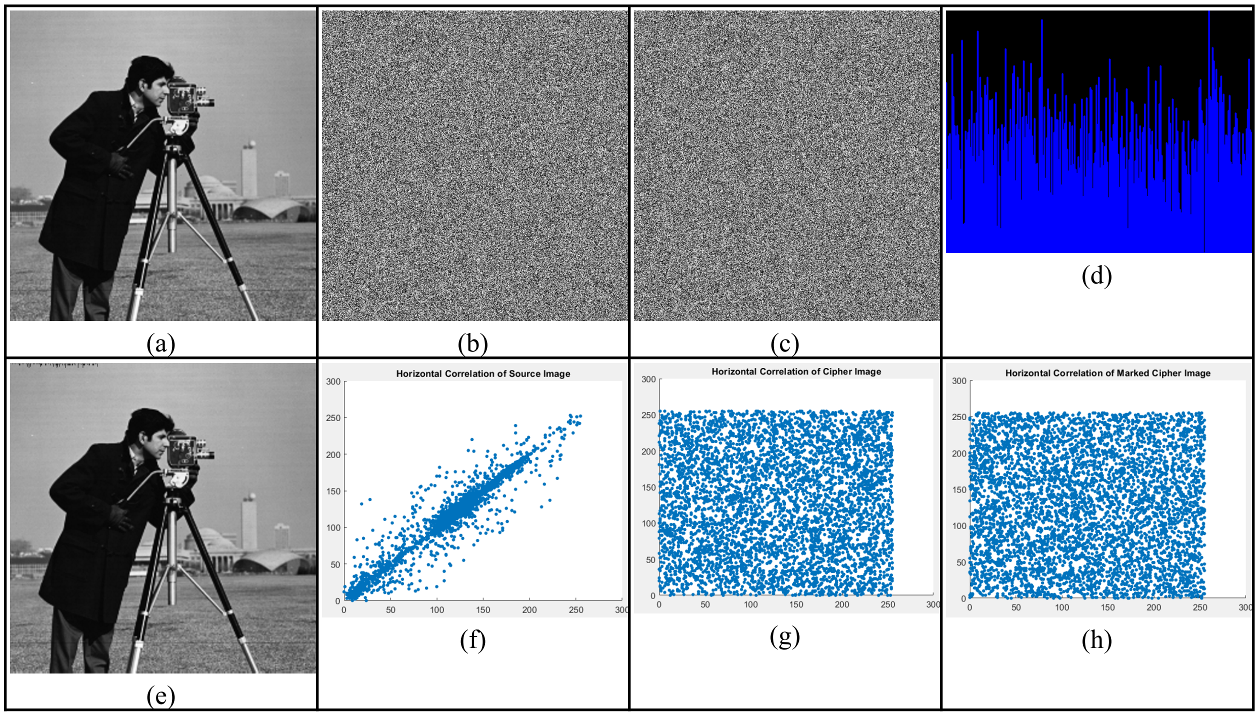

Fig 2. (a) source image; (b) ciphertext image before embedding; (c) ciphertext image after embedding; (d) histogram of the pixel value distribution in the ciphertext embedded with data; (e) reconstructed image;

(f) horizontal correlation of source image; (g) horizontal correlation of ciphertext image; g) horizontal correlation of ciphertext image after embedding data.

As evident, both the horizontal and vertical correlation are small compared to their original images.

The metrics NPCR and UACI are designed to test for "number of changing pixels" and "number of averaged changed intensity", respectively, between two ciphertexts originating from the same original image except with a small difference (such as one single pixel being different). Here that pixel is chosen to be (128, 128), the encryption key - that is, parameters for the chaotic PRNG - are the same for the generation of the two ciphertext images, each started from its initial state.[1] The two metrics are defined as thus:

$$D(i,j) = \begin{cases} 0, if C^1(i,j) = C^2(i,j) \\ 1, if C^1(i,j) \neq C^2(i,j) \end{cases} \quad (1)$$

$$\text{NPCR: } \mathcal{N}(C^1, C^2) = \sum_{i,j} \frac{D(i,j)}{T} \times 100\% \quad (2)$$

$$\text{UACI: } \mathcal{U}(C^1, C^2) = \sum_{i,j} \frac{|C^1(i,j) - C^2(i,j)|}{F \cdot T} \times 100\% \quad (3)$$

[5]

where C^1, C^2 are the two ciphertexts generated with a pixel difference in their original source image, T the total number of pixels in the image, and F the largest value supported by a pixel (255 in grayscale). The authors in [5] investigated the critical values for both NPCR and UACI should be in order for the probability of rejecting the null hypothesis, that the two ciphertexts are "not random-like", when they in fact are, is α; for NPCR, the test is if $NPCR < N^*_\alpha$ where N^*_α is the critical value to reject the null hypothesis with α-level significance; for UACI, the test is whether $UACI \in (U^{*-}_\alpha, U^{*+}_\alpha)$, rejecting the null hypothesis like the test for NPCR otherwise. These critical values are calculated based on their modelling with the parameters of image size ($M \times N$) and Fon the two metrics. Values for $\alpha = 0.05, 0.01, 0.001$ are calculated and relevant metrics are obtained from [5] and shown below. [5]

$N^*_{0.05}$	$N^*_{0.01}$	$N^*_{0.001}$
99.5893%	99.5810%	99.5717%

Table II. Subset of the numerical results for NPCR randomness test from [5] for $M \times N = 512 \times 512$ and $F = 255$.

[1] It was unclear to me whether or not the second ciphertext C^2 should continue using the same chaotic PRNG or restarted with the same parameters; here I simply assumed the latter instead.

$U^{*-}_{0.05}/U^{*+}_{0.05}$	$U^{*-}_{0.01}/U^{*+}_{0.01}$	$U^{*-}_{0.001}/U^{*+}_{0.001}$
33.3730% / 33.5541%	33.3445% / 33.5826%	33.3115% / 33.6156%

Table III. Subset of the numerical results for UACI randomness test from [5] for $M \times N = 512 \times 512$ and $F = 255$.

Interestingly enough, the definition provided in [1] regarding NPCR for $D(i, j)$ is the opposite of that reported in [5], and furthermore, in [1] both metrics were seemingly conducted between the source image and the ciphertext image, instead of being computed between two ciphertexts. [1] As such, assuming that the definition of $D(i, j)$ is but a typo, then perhaps the nature of NPCR and UACI used here is simply as metrics for measuring how different the source and ciphertext images simply are. As such, the results when adjusted as thus are presented in the square bracket values, $[x_1 \ / \ x_2]$ with x_1 for unmarked ciphertext and x_2 the marked ciphertext. For NPCR, both Lena and Cameraman images passed all three significance tests; for UACI tests, neither lied within the defined intervals.

The metrics PSNR and SSIM are used for evaluating the quality of reconstructed images, that is, its similarity to the original image. PSNR is the "logarithm of the mean square error (MSE) of an image", whereby the value approaches infinity for identical images, and the lower it is the more differences exist between the compared images. SSIM on the other hand is based around the comparison of three factors in the input images: difference in luminance, contrast and structure, wherein the structural factor compares the correlation coefficients of the two images; its ideal value is 1. [6] From table I we have PSNR of around ~37s and SSIM of close to ~1.

References

[1] P. Puteaux and W. Puech, "An Efficient MSB Prediction-Based Method for High-Capacity Reversible Data Hiding in Encrypted Images," in IEEE Transactions on Information Forensics and Security, vol. 13, no. 7, pp. 1670-1681, July 2018, doi: 10.1109/TIFS.2018.2799381.

[2] Shujun L., Xuanqin M., Yuanlong C. (2001) Pseudo-random Bit Generator Based on Couple Chaotic Systems and Its Applications in Stream-Cipher Cryptography. In: Rangan C.P., Ding C. (eds) Progress in Cryptology — INDOCRYPT 2001. INDOCRYPT 2001. Lecture Notes in Computer Science, vol 2247. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45311-3_30

[3] S. Li, G. Chen, and X. Mou, "On the dynamical degradation of digital piecewise linear chaotic maps," International Journal of Bifurcation and Chaos, vol. 15, no. 10, pp. 3119–3151, 2005.

[4] Li S., Li Q., Li W., Mou X., Cai Y. (2001) Statistical Properties of Digital Piecewise Linear Chaotic Maps and Their Roles in Cryptography and Pseudo-Random Coding. In: Honary B.

(eds) Cryptography and Coding. Cryptography and Coding 2001. Lecture Notes in Computer Science, vol 2260. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45325-3_19

[5] Y. Wu, J. Noonan and S. Agaian, “Npcr and Uaci Randomness Tests for Image Encryption,” Cyber Journals: Multidisciplinary, Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), 2011, pp. 31-38.

[6] Setiadi, D.I.M. PSNR vs SSIM: imperceptibility quality assessment for image steganography. Multimed Tools Appl 80, 8423–8444 (2021). https://doi.org/10.1007/s11042-020-10035-z