INTERNATIONAL JOURNAL OF MERGING TECHNOLOGY AND ADVANCED RESEARCH IN COMPUTING

IJMTARC – VOLUME – VI – ISSUE – 24, OCT-DEC, 2018

ISSN: 2320-1363

Public Key Cryptography Based Lossless and Reversible Data Hiding in Encrypted Images

1.V.Suryanarayana Adireddi ,PG Scholar Dept of ECE Eswar college of engineering Narasaraopet.

2.Sk.Lal John Basha,Asst prof Dept of ECE Eswar college of engineering Narasaraopet

ABSTRACT:

This paper proposes a lossless, a reversible, and a combined data hiding schemes for ciphertext images encrypted by public key cryptosystems with probabilistic and homomorphic properties. In the lossless scheme, the ciphertext pixels are replaced with new values to embed the additional data into several LSB-planes of ciphertext pixels by multi-layer wet paper coding. Then, the embedded data can be directly extracted from the encrypted domain, and the data embedding operation does not affect the decryption of original plaintext image. In the reversible scheme, a preprocessing is employed to shrink the image histogram before image encryption, so that the modification on encrypted images for data embedding will not cause any pixel oversaturation in plaintext domain. Although a slight distortion is introduced, the embedded data can be extracted and the original image can be recovered from the directly decrypted image. Due to the compatibility between the lossless and reversible schemes, the data embedding operations in the two manners can be simultaneously performed in an encrypted image. With the combined technique, a receiver may extract a part of embedded data before decryption, and extract another part of embedded data and recover the original plaintext image after decryption.

I.INTRODUCTION

Visual cryptography is a cryptographic technique which allows visual information (pictures, text, etc.) to be encrypted in such a way that decryption becomes a mechanical operation that does not require a computer.One of the best-known techniques has been credited to Moni Naor and Adi Shamir. They demonstrated a visual secret sharing scheme, where an image was broken up into n shares so that only someone with all n shares could decrypt the image, while any n − 1 shares revealed no information about the original image. Each share was printed on a separate transparency, and decryption was performed by overlaying the shares. When all n shares were overlaid, the original image would appear.

Using a similar idea, transparencies can be used to implement a one-time pad encryption, where one transparency is a shared random pad, and another transparency acts as the cipher text.In cryptography, encryption is the process of encoding messages (or information) in such a way that eavesdroppers or hackers cannot read it, but that authorized parties can. In an encryption scheme, the message or information (referred to as plaintext) is encrypted using an encryption algorithm, turning it into an unreadable ciphertext (ibid.). This is usually done with the use of an encryption key, which specifies how the message is to be encoded. Any adversary that can see the ciphertext should not be able to determine anything about the original message. An authorized party, however, is able to decode the ciphertext using a decryption algorithm that usually requires a secret decryption key that adversaries do not have access to. For technical reasons, an encryption scheme usually needs a key-generation algorithm to randomly produce keys.

Encryption is also used to protect data in transit, for example data being transferred via networks (e.g. the Internet, e-commerce), mobile telephones, wireless microphones, wireless intercom systems, Bluetooth devices and bank automatic teller machines. There have been numerous reports of data in transit being intercepted in recent years. Encrypting data in transit also helps to secure it as it is often difficult to physically secure all access to networks.

Encryption, by itself, can protect the confidentiality of messages, but other techniques are still needed to

IJMTARC – VOLUME – VI – ISSUE – 24, OCT-DEC, 2018

ISSN: 2320-1363

protect the integrity and authenticity of a message; for example, verification of a message authentication code (MAC) or a digital signature. Standards and cryptographic software and hardware to perform encryption are widely available, but successfully using encryption to ensure security may be a challenging problem. A single slip-up in system design or execution can allow successful attacks. Sometimes an adversary can obtain unencrypted information without directly undoing the encryption. See e.g., traffic analysis, TEMPEST, or Trojan horse.

One of the earliest public key encryption applications was called Pretty Good Privacy (PGP). Digital signature and encryption must be applied at message creation time (i.e. on the same device it has been composed) to avoid tampering. Otherwise any node between the sender and the encryption agent could potentially tamper it. It should be noted that encrypting at the time of creation only adds security if the encryption device itself has not been tampered with.

Most of the work on reversible data hiding focuses on the data embedding/extracting on the plain spatial domain. But, in some applications, an inferior assistant or a channel administrator hopes to append some additional message, such as the origin information, image notation or authentication data, within the encrypted image though he does not know the original image content. And it is also hopeful that the original content should be recovered without any error after image decryption and message extraction at receiver side. Reference presents a practical scheme satisfying the above-mentioned requirements. A content owner encrypts the original image using an encryption key, and a data-hider can embed additional data into the encrypted image using a data-hiding key though he does not know the original content. With an encrypted image containing additional data, a receiver may first decrypt it according to the encryption key, and then extract the embedded data and recover the original image according to the data-hiding key. In the scheme, the data extraction is not separable from the content.

The proposed scheme is made up of image encryption, data embedding and data-extraction/image-recovery phases. In our paper we implement the key generation by using hill chipher method. In our project, we give additional password to decrypt our image. The content owner encrypts the original uncompressed image using an encryption key to produce an encrypted image. Then, the data-hider compresses the least significant bits (LSB) of the encrypted image using a data-hiding key to create a sparse space to accommodate the additional data. At the receiver side, the data embedded in the created space can be easily retrieved from the encrypted image containing additional data according to the data-hiding key. Since the data embedding only affects the LSB, a decryption with the encryption key can result in an image similar to the original version. When using both of the encryption and data-hiding keys, the embedded additional data can be successfully extracted and the original image can be perfectly recovered by exploiting the spatial correlation in natural image.

II.EXISTING SYSTEM

Cryptography is an art of protecting the information by transforming into an unreadable and untraceable format known as cipher text. Only the person who possess the secret key can decipher or we can say decrypt the message into the original form. Cryptography is the technique by which one can send and share the information in a secret manner. Due the cryptography the information seems to be appearing like a garbage value and it is always almost impossible to find the information content lying under the image or a video file. The information looks like hidden inside the image or the video file. A very simplest and well known algorithm for cryptography is as shown in fig 2. The encryption key generator is used to generate the encryption key as well as the public key as shown in the below block diagram. By using the encryption key the information content to be sent gets encrypted by the encryptor. The encrypted information is then transmitted to the particular receiver. At the receiver end the cryptography decryptor is used which extracts the original information content mapped onto the image or a video file with the help of a public key provided by the transmitter section. By the use of the cryptography method only, the receiver which has the knowledge of the public key can retrieve the original information content from the image or a video file. So even if any unwanted person or a source gets the image or a video file with information content hidden in it, it cannot be extracted without proper public key. So public key plays a vital role in the whole cryptography process.

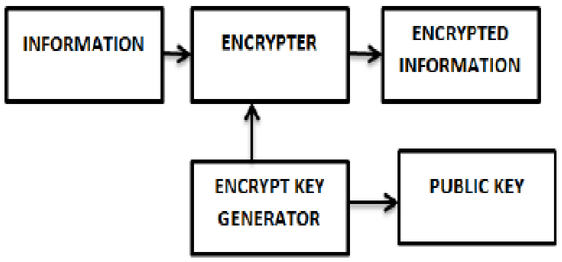


Fig.1. Cryptography encrypter

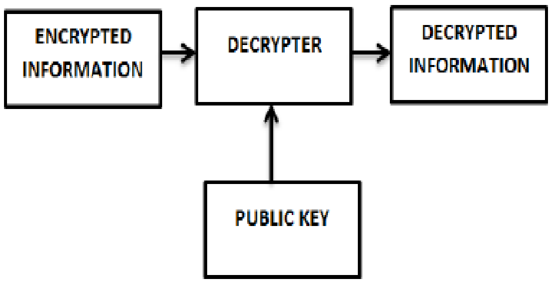


Fig..2.Cryptography decrypter

Technically in simple words "cryptographymeans hiding one piece of data within another". Modern cryptographyuses the opportunity of hiding information into digital multimedia files and also at the network packet level. Hiding information into a media requires following elements. The cover media(C) that will hold the hidden data

- The secret message (M), may be plain text, cipher text or any type of data
- The stego function (Fe) and its inverse (Fe-1)
- An optional stego-key (K) or password may be used to hide and unhide the message .

Cryptography is the art or study of hiding information by inserting secret messages in other messages. Medium where information is inserted can be anything. This medium is called the cover object. Cryptography that is applied to hide information on the cover of digital objects is called Digital Cryptography. Cover objects that are used in digital cryptography can vary, for example in the image archive. Cryptography algorithms in the image archive have been widely developed. Meanwhile, cryptography algorithms in audio archive are relatively few. In recent years there are so many algorithm have been developed to provide more security, enhanced quality with easy implementation and faster calculations. Among them all of the techniques have their own drawbacks like computational complexity, time consumption and reconstruction of secret information etc., Here, in this proposal we implemented pixel mapping based video steganography, which is a very simple and easy calculations and also provide more security.

III.PROPOSED SYSTEM

This paper proposes a lossless, a reversible, and a combined data hiding schemes for public-key-encrypted images by exploiting the probabilistic and homomorphic properties of cryptosystems. With these schemes, the pixel division/reorganization is avoided and the encryption/decryption is performed on the cover pixels directly, so that the amount of encrypted data and the computational complexity are lowered. In the lossless scheme, due to the probabilistic property, although the data of encrypted image are modified for data embedding, a direct decryption can still result in the original plaintext image while the embedded data can be extracted in the encrypted domain. In the reversible scheme, a histogram shrink is realized before encryption so that the modification on encrypted image for data embedding does not cause any pixel oversaturation in plaintext domain. Although the data embedding on encrypted domain may result in a slight distortion in plaintext domain due to the homomorphic property, the embedded data can be extracted and the original content can be recovered from the directly decrypted image. Furthermore, the data embedding operations of the lossless and the reversible schemes can be simultaneously performed in an encrypted image. With the combined technique, a receiver may extract a part of embedded data before decryption, and extract another part of embedded data and recover the original plaintext image after decryption

IV.LOSSLESS DATA HIDING SCHEME

In this section, a lossless data hiding scheme for public- key-encrypted images is proposed. There are three parties in the scheme: an image provider, a data-hider, and a receiver. With a cryptosystem possessing probabilistic property, the image provider encrypts each pixel of the original plaintext image using the public key of the receiver, and a data-hider

IJMTARC – VOLUME – VI – ISSUE – 24, OCT-DEC, 2018 ISSN: 2320-1363

who does not know the original image can modify the ciphertext pixel-values to embed some additional data into the encrypted image by multi-layer wet paper coding under a condition that the decrypted values of new and original cipher-text pixel values must be same. When having the encrypted image containing the additional data, a receiver knowing the data hiding key may extract the embedded data, while a receiver with the private key of the cryptosystem may perform decryption to retrieve the original plaintext image. In other words, the embedded data can be extracted in the encrypted domain, and cannot be extracted after decryption since the decrypted image would be same as the original plaintext image due to the probabilistic property. That also means the data embedding does not affect the decryption of the plaintext image. The sketch of lossless data hiding scheme is shown in Figure 1.

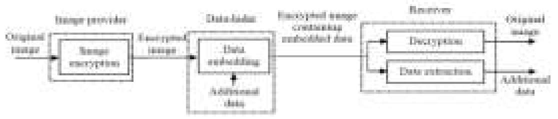


Figure.3. Sketch of lossless data hiding scheme for public-key-encrypted images

V. IMAGE ENCRYPTION

In this phase, the image provider encrypts a plaintext image using the public key of probabilistic cryptosystem *pk*. For each pixel value *m*(*i*, *j*) where (*i*, *j*) indicates the pixel position, the image provider calculates its ciphertext value

$c(i,j)= E[p_k, m(i,j), r(i,j)]$

Where *E* is the encryption operation and *r*(*i*, *j*) is a random value. Then, the image provider collects the ciphertext values of all pixels to form an encrypted image. Actually, the proposed scheme is capitable with various probabilistic public-key cryptosystems, such as Paillier and Damgard-Jurik cryptosystems . With Paillier cryptosystem , for two large primes *p* and *q*, calculate $n = p\cdot q$, $\lambda = \text{lcm}(p-1, q-1)$, where lcm means the least common multiple. Here, it should meet that $\gcd(n, (p-1)\cdot(q-1)) = 1$, where gcd means the greatest common divisor. The public key is composed of *n* and a randomly selected integer *g* in , while the private key is composed of λ and *Z*

$$\mu = \left(L\left(g^{\lambda} \bmod n^2\right)\right)^{-1} \bmod n$$

Where

$$L(x) = \frac{(x-1)}{n}$$

In this case, (1) implies

$$c(i,j)= g^{m(i,j)} \cdot (r(i,j))^{n} \mod n^2$$

Where *r*(*i*, *j*) is a random integer in *Z*n*. The plaintext pixel value can be obtained using the private key,

$$m(i,j) = L\left(\left(c(i,j)\right)^{\lambda} \bmod n^2\right)\cdot \mu \bmod n$$

As a generalization of Paillier cryptosystem, Damgard-Jurik cryptosystem [25] can be also used to encrypt the plaintext image. Here, the public key is composed of *n* and an element *g* in such that *g* = (1+*n*)*j*·*x*mod *ns*+1 for a known *j* relatively prime to *n* and *x* belongs to a group isomorphic to *Z*n*, and we may choose *d* as the private key when meeting *d* mod *n* ∈*Z*n* and *d* = 0 mod λ. Then, the encryption in (1) can be rewritten as

$$c(i,j) = g^{m(i,j)}\cdot(r(i,j))^{n^s} \bmod n^{s+1}$$

Where *r*(*i*, *j*) is a random integer in . By applying a recursive version of Paillier decryption, the plaintext value can be obtained from the ciphertext value using the private key. Note that, because of the probabilistic property of the two cryptosystems, the same gray values at different positions may correspond to different ciphertext values

5.1 DATA EMBEDDING

When having the encrypted image, the data-hider may embed some additional data into it in a lossless manner. The pixels in the encrypted image are reorganized as a sequence according to the data hiding key. For each encrypted pixel, the data-hider selects a random integer *r'*(*i*, *j*) in *Z*n* and calculates

$$c'(i,j) = c(i,j)\cdot(r'(i,j))^{n} \bmod n^2$$

if Paillier cryptosystem is used for image encryption, while the data-hider selects a random integer *r'*(*i*, *j*) in and calculates

$$c'(i,j) = c(i,j)\cdot(r'(i,j))^{n^s} \bmod n^{s+1}$$

if Damgard-Jurik cryptosystem is used for image encryption. We denote the binary representations of *c*(*i*, *j*) and *c'*(*i*, *j*) as *bk*(*i*, *j*) and *b'k*(*i*, *j*), respectively,

IJMTARC – VOLUME – VI – ISSUE – 24, OCT-DEC, 2018 ISSN: 2320-1363

$$b_k(i,j) = \left\lfloor \frac{c(i,j)}{2^{k-1}} \right\rfloor \bmod 2, \qquad k = 1, 2, \ldots \qquad (9)$$

$$b'_k(i,j) = \left\lfloor \frac{c'(i,j)}{2^{k-1}} \right\rfloor \bmod 2, \qquad k = 1, 2, \ldots \qquad (10)$$

By viewing the *k*-th LSB of encrypted pixels as a wet paper channel (WPC) and the *k*-th LSB in *Sk*as "dry" elements of the wet paper channel, the data-hider may employ the wet paper coding to embed the additional data by replacing a part of *c*(*i*, *j*) with *c'*(*i*, *j*). The details will be given in the following.

Considering the first LSB, if *c*(*i*, *j*) are replaced with *c'*(*i*, *j*), the first LSB in *S*1 would be flipped and the rest first LSB would be unchanged. So, the first LSB of the encrypted pixels can be regarded as a WPC, which includes changeable (dry) elements and unchangeable (wet) elements. In other words, the first LSB in *S*1 are dry elements and the rest first LSB are wet positions. By using the wet paper coding, one can represent on average *Nd*bits by only flipping a part of dry elements where *Nd*is the number of dry elements. In this scenario, the data-hider may flip the dry elements by replacing *c*(*i*, *j*) with *c'*(*i*, *j*). Denoting the number of pixels in the image as *N*, the data-hider may embed on average *N*/2 bits in the first LSB-layer using wet paper coding.

Considering the second LSB (SLSB) layer, we call the SLSB in *S*2 as dry elements and the rest SLSB as wet elements. Note that the first LSB of ciphertext pixels in *S*1 have been determined by replacing *c*(*i*, *j*) with *c'*(*i*, *j*) or keeping *c*(*i*, *j*) unchanged in the first LSB-layer embedding, meaning that the SLSB in *S*1 are unchangeable in the second layer. Then, the data-hider may flip a part of SLSB in *S*2 by replacing *c*(*i*, *j*) with *c'*(*i*, *j*) to embed on average *N*/4 bits using wet paper coding.

Similarly, in the *k*-th LSB layer, the data-hider may flip a part of *k*-th LSB in *Sk*to embed on average *N*/2*k* bits. When the data embedding is implemented in *K* layers, the total *N*·(1−1/2*K*) bits, on average, are embedded. That implies the embedding rate, a ratio between the number of embedded bits and the number of pixels in cover image, is approximately (1−1/2*K*). That implies the upper bound of the embedding rate is 1 bit per pixel. The next subsection will show that, although a part of *c*(*i*, *j*) is replaced with *c'*(*i*, *j*), the original plaintext image can still be obtained by decryption.

5.2 DATA EXTRACTION AND IMAGE DECRYPTION

After receiving an encrypted image containing the additional data, if the receiver knows the data-hiding key, he may calculate the *k*-th LSB of encrypted pixels, and then extract the embedded data from the *K* LSB-layers using wet paper coding. On the other hand, if the receiver knows the private key of the used cryptosystem, he may perform decryption to obtain the original plaintext image. When Paillier cryptosystem is used, Equation (4) implies

$$c(i,j) = g^{m(i,j)} \cdot (r(i,j))^n + \alpha \cdot n^2$$

Where α is an integer. By substituting (12) into , there is

$$c'(i,j) = g^{m(i,j)} \cdot (r(i,j) \cdot r'(i,j))^n \bmod n^2$$

Since *r*(*i*, *j*)·*r'*(*i*, *j*) can be viewed as another random integer in *Z*n*, the decryption on *c'*(*i*, *j*) will result in the plaintext value

$$m(i,j) = L\left((c'(i,j))^{\lambda} \bmod n^2\right) \cdot \mu \bmod n$$

Similarly, when Damgard-Jurik cryptosystem is used,

$$c'(i,j) = g^{m(i,j)} \cdot (r(i,j) \cdot r'(i,j))^{n^s} \bmod n^{s+1}$$

The decryption on *c'*(*i*, *j*) will also result in the plaintext value. In other words, the replacement of ciphertext pixel values for data embedding does not affect the decryption result.

5.3 REVERSIBLE DATA HIDING SCHEME

This section proposes a reversible data hiding scheme for public-key-encrypted images. In the reversible scheme, a preprocessing is employed to shrink the image histogram, and then each pixel is encrypted with additive homomorphic cryptosystem by the image provider. When having the encrypted image, the data-hider modifies the ciphertext pixel values to embed a bit-sequence generated from the additional data and error-correction codes. Due to the homomorphic property, the modification in encrypted domain will result in slight increase/decrease on plaintext pixel values, implying that a decryption can be implemented to obtain an image similar to the original plaintext image on receiver side. Because of the histogram shrink before encryption, the data embedding operation does not cause any overflow/underflow in the directly decrypted image. Then, the original plaintext image can be recovered and the embedded additional data can be extracted from the directly decrypted image. Note that the data-

IJMTARC – VOLUME – VI – ISSUE – 24, OCT-DEC, 2018

ISSN: 2320-1363

extraction and content-recovery of the reversible scheme are performed in plaintext domain, while the data extraction of the previous lossless scheme is performed in encrypted domain and the content recovery is needless. The sketch of reversible data hiding scheme is given in Figure 2.

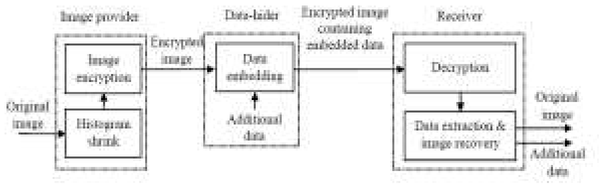


Figure.5.2. Sketch of reversible data hiding scheme for public-key-encrypted images

5.4 HISTOGRAM SHRINK AND IMAGE ENCRYPTION

In the reversible scheme, a small integer δ shared by the image provider, the data-hider and the receiver will be used, and its value will be discussed later. Denote the number of pixels in the original plaintext image with gray value *v* as *hv*, implying

$$\sum_{v=0}^{255} h_v = N$$

where *N* is the number of all pixels in the image. The image provider collects the pixels with gray values in [0, δ+1], and represent their values as a binary stream BS1. When an efficient lossless source coding is used, the length of BS1

$$l_1 \approx \sum_{v=0}^{\delta+1} h_v \cdot H\left(\frac{h_0}{\sum_{v=0}^{\delta+1} h_v}, \frac{h_1}{\sum_{v=0}^{\delta+1} h_v}, \ldots, \frac{h_{\delta+1}}{\sum_{v=0}^{\delta+1} h_v} \right)$$

Where *H*(·) is the entropy function. The image provider also collects the pixels with gray values in [255−δ, 255], and represent their values as a binary stream BS2 with a length *l*2. Similarly,

$$l_2 \approx \sum_{v=255-\delta}^{255} h_v \cdot H\left(\frac{h_{255-\delta}}{\sum_{v=255-\delta}^{255} h_v}, \frac{h_{255-\delta+1}}{\sum_{v=255-\delta}^{255} h_v}, \ldots, \frac{h_{255}}{\sum_{v=255-\delta}^{255} h_v} \right)$$

Then, the gray values of all pixels are enforced into [δ+1, 255−δ],

$$m_S(i,j) = \begin{cases} 255-\delta & , \text{ if } m(i,j) \geq 255-\delta \\ m(i,j) & , \text{ if } \delta+1 < m(i,j) < 255-\delta \\ \delta+1 & , \text{ if } m(i,j) \leq \delta+1 \end{cases}$$

Denoting the new histogram as *h'v*, there must be

$$h'_v = \begin{cases} 0 & , \quad v \leq \delta \\ \sum_{v=0}^{\delta+1} h_v & , \quad v = \delta+1 \\ h_v & , \quad \delta+1 < v < 255-\delta \\ \sum_{v=255-\delta}^{255} h_v & , \quad v = 255-\delta \\ 0 & , \quad v > 255-\delta \end{cases}$$

The image provider finds the peak of the new histogram

$$V = \arg\max_{\delta+1 \leq v \leq 255-\delta} h'_v$$

The image provider also divides all pixels into two sets: the first set including (*N*−8) pixels and the second set including the rest 8 pixels, and maps each bit of BS1, BS2 and the LSB of pixels in the second set to a pixel in the first set with gray value *V*. Since the gray values close to extreme black/white are rare, there is

$$h'_V \geq l_1 + l_2 + 16$$

When δ is not too large, In this case, the mapping operation is feasible. Here, 8 pixels in the second set cannot be used to carry BS1/BS2 since their LSB should be used to carry the value of *V*, while 8 pixels in the first set cannot be used to carry BS1/BS2 since their LSB should be used to carry the original LSB of the second set. So, a total of 16 pixels cannot be used for carrying BS1/BS2. That is the reason that there is a value 16 in (22). The experimental result on 1000 natural images shows is always right when δ is less than 15. So, we recommend the parameter δ < 15. Then, a histogram shift operation is made

$$m_T(i,j) = \begin{cases} m_S(i,j) & , \text{ if } m_S(i,j) > V \\ V & , \text{ if } m_S(i,j) = V \text{ and the corresponding bit is 0} \\ V-1 & , \text{ if } m_S(i,j) = V \text{ and the corresponding bit is 1} \\ m_S(i,j)-1 & , \text{ if } m_S(i,j) < V \end{cases}$$

That means the value of *V* is embedded into the LSB of the second set. This way, all pixel values must fall into [δ, 255−δ]. At last, the image provider encrypts all pixels using a public key cryptosystem with additive homomorphic property, such as Paillier and Damgard-Jurik cryptosystems. When Paillier cryptosystem is used, the ciphertext pixel is

$$c(i,j) = g^{m_T(i,j)} \cdot (r(i,j))^n \bmod n^2$$

And, when Damgard-Jurik cryptosystem is used, the ciphertext pixel is

$$c(i,j) = g^{m_T(i,j)} \cdot (r(i,j))^{n^s} \bmod n^{s+1}$$

Then, the ciphertext values of all pixels are collected to form an encrypted image.

5.5 DATA EMBEDDING

When having the encrypted image, the data-hider may embed some additional data into it in a lossless manner. The pixels in the encrypted image are reorganized as a sequence according to the data hiding key. For each encrypted pixel, the data-hider selects a random integer *r'*(*i*, *j*) in *Z*n* and calculates

$$c'(i,j) = c(i,j) \cdot (r'(i,j))^n \bmod n^2$$

if Paillier cryptosystem is used for image encryption, while the data-hider selects a random integer *r'*(*i*, *j*) in and calculates

$$c'(i,j) = c(i,j) \cdot (r'(i,j))^{n^s} \bmod n^{s+1}$$

if Damgard-Jurik cryptosystem is used for image encryption. We denote the binary representations of *c*(*i*, *j*) and *c'*(*i*, *j*) as *bk*(*i*, *j*) and *b'k*(*i*, *j*), respectively,

$$b_k(i,j) = \left\lfloor \frac{c(i,j)}{2^{k-1}} \right\rfloor \bmod 2, \qquad k = 1, 2, \ldots \qquad (9)$$

$$b'_k(i,j) = \left\lfloor \frac{c'(i,j)}{2^{k-1}} \right\rfloor \bmod 2, \qquad k = 1, 2, \ldots \qquad (10)$$

By viewing the *k*-th LSB of encrypted pixels as a wet paper channel (WPC) [26] and the *k*-th LSB in *Sk*as "dry" elements of the wet paper channel, the data-hider may employ the wet paper coding [26] to embed the additional data by replacing a part of *c*(*i*, *j*) with *c'*(*i*, *j*). The details will be given in the following.

Considering the first LSB, if *c*(*i*, *j*) are replaced with *c'*(*i*, *j*), the first LSB in *S*1 would be flipped and the rest first LSB would be unchanged. So, the first LSB of the encrypted pixels can be regarded as a WPC, which includes changeable (dry) elements and unchangeable (wet) elements. In other words, the first LSB in *S*1 are dry elements and the rest first LSB are wet positions. By using the wet paper coding, one can represent on average *Nd*bits by only flipping a part of dry elements where *Nd*is the number of dry elements. In this scenario, the data-hider may flip the dry elements by replacing *c*(*i*, *j*) with *c'*(*i*, *j*). Denoting the number of pixels in the image as *N*, the data-hider may embed on average *N*/2 bits in the first LSB-layer using wet paper coding.

Considering the second LSB (SLSB) layer, we call the SLSB in *S*2 as dry elements and the rest SLSB as wet elements. Note that the first LSB of ciphertext pixels in *S*1 have been determined by replacing *c*(*i*, *j*) with *c'*(*i*, *j*) or keeping *c*(*i*, *j*) unchanged in the first LSB-layer embedding, meaning that the SLSB in *S*1 are unchangeable in the second layer. Then, the data-hider may flip a part of SLSB in *S*2 by replacing *c*(*i*, *j*) with *c'*(*i*, *j*) to embed on average *N*/4 bits using wet paper coding.

5.6 IMAGE DECRYPTION, DATA EXTRACTION AND CONTENT RECOVERY

After receiving an encrypted image containing additional data, the receiver firstly performs decryption using his private key. We denote the decrypted pixels as *m'*(*i*, *j*). Due to the homomorphic property, the decrypted pixel values in Set A meet

$$m'(i,j) = \begin{cases} m_T(i,j) + \delta, & \text{if the corresponding bit is 1} \\ m_T(i,j) - \delta, & \text{if the corresponding bit is 0} \end{cases}$$

On the other hand, the decrypted pixel values in Set B are just *m*T(*i*, *j*) since their ciphertext values are unchanged in data embedding phase. When δ is small, the decrypted image is perceptually similar to the original plaintext image. Then, the receiver with the data-hiding key can extract the embedded data from the directly decrypted image.

$$\overline{m}_T(i,j) = \frac{m_T(i-1,j) + m_T(i,j-1) + m_T(i+1,j) + m_T(i,j+1)}{4}$$

He estimates the pixel values in Set A using their neighbors the receiver may employ the error-correction method to retrieve the original coded bit-sequence and the embedded additional data. Note that, with a larger δ, the error rate in the estimate of coded bits would be lower, so that more additional data can be embedded when ensuring successful error correction and data extraction. In other words, a smaller δ would result in a higher error rate in the estimate of coded bits, so that the error correction may be unsuccessful when excessive payload is embedded. That means the embedding capacity of the reversible data hiding scheme is depended on the value of δ. After retrieving the original coded bit-sequence and the embedded additional data, the original plaintext image may be further recovered. For the pixels in Set A, *m*T(*i*, *j*) are retrieved according to the coded bit-sequence,

$$m_T(i,j)=\begin{cases} m'(i,j)-\delta & , \text{ if the corresponding bit is 1} \\ m'(i,j)+\delta & , \text{ if the corresponding bit is 0} \end{cases}$$

For the pixels in Set B, as mentioned above, *m*T(*i*, *j*) are just *m'*(*i*, *j*). Then, divides all *m*T(*i*, *j*) into two sets: the first one including (*N*−8) pixels and the second one including the rest 8 pixels. The receiver may obtain the value of *V* from the LSB in the second set, and retrieve *m*S(*i*, *j*) of the first set

$$m_S(i,j)=\begin{cases} m_T(i,j) & , \text{ if } m_T(i,j)>V \\ V & , \text{ if } m_T(i,j)=V \text{ or } V-1 \\ m_T(i,j)+1 & , \text{ if } m_T(i,j)<V-1 \end{cases}$$

The receiver extracts a bit 0 from a pixel with *m*T(*i*, *j*) = *V* and a bit 1 from a pixel with *m*T(*i*, *j*) = *V*−1. After decomposing the extracted data into BS1, BS2 and the LSB of *m*S(*i*, *j*) in the second set, the receiver retrieves *m*S(*i*, *j*) of the second set,

$$m_S(i,j)=\begin{cases} m_T(i,j) & , \text{ if LSB of } m_S(i,j) \text{ and } m_T(i,j) \text{ are} \\ m_T(i,j)+1 & , \text{ if LSB of } m_S(i,j) \text{ and } m_T(i,j) \text{ are} \end{cases}$$

Collect all pixels with *m*S(*i*, *j*) = δ+1, and, according to BS1, recover their original values within [0, δ+1]. Similarly, the original values of pixels with *m*S(*i*, *j*) = 255−δ are recovered within [255−δ, 255] according to BS2. This way, the original plaintext image is recovered

5.7 COMBINED DATA HIDING SCHEME

As described in Sections 3 and 4, a lossless and a reversible data hiding schemes for public-key-encrypted images are proposed. In both of the two schemes, the data embedding operations are performed in encrypted domain. On the other hand, the data extraction procedures of the two schemes are very different. With the lossless scheme, data embedding does not affect the plaintext content and data extraction is also performed in encrypted domain. With the reversible scheme, there is slight distortion in directly decrypted image caused by data embedding, and data extraction and image recovery must be performed in plaintext domain. That implies, on receiver side, the additional data embedded by the lossless scheme cannot be extracted after decryption, while the additional data embedded by the reversible scheme cannot extracted before decryption. In this section, we combine the lossless and reversible schemes to construct a new scheme, in which data extraction in either of the two domains is feasible. That means the additional data for various purposes may be embedded into an encrypted image, and a part of the additional data can be extracted before decryption and another part can be extracted after decryption.

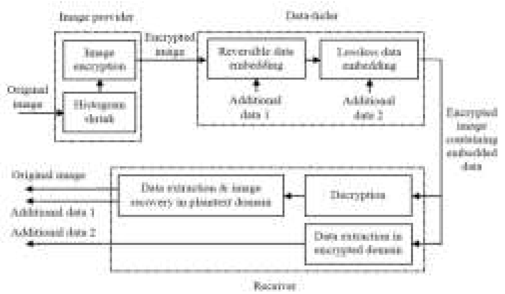


Figure.4. Sketch of combined scheme

In the combined scheme, the image provider performs histogram shrink and image encryption as described in Subsection 3.A. When having the encrypted image, the data-hider may embed the first part of additional data using the method described in Subsection . Denoting the ciphertext pixel values containing the first part of additional data as *c'*(*i*, *j*), the data-hider calculates

$$c''(i,j)=c'(i,j)\cdot\left(r''(i,j)\right)^{n} \bmod n^2$$

$$c''(i,j)=c'(i,j)\cdot\left(r''(i,j)\right)^{n^s} \bmod n^{s+1}$$

where *r''*(*i*, *j*) are randomly selected in *Z*n* or for Paillier and Damgard-Jurik cryptosystems, respectively. Then, he may employ wet paper coding in several LSB-planes of ciphertext pixel values to

IJMTARC – VOLUME – VI – ISSUE – 24, OCT-DEC, 2018

ISSN: 2320-1363

embed the second part of additional data by replacing a part of $c'(i, j)$ with $c''(i, j)$.

VI.MODULES AND MODULE DESCRIPTION

MODULES

- Input image initialization,
- Image Encryption,
- Data Embedding,
- Data Extraction and Image Recovery,
- Compute PSNR.

MODULE DESCRIPTION

6.1. INPUT IMAGE INITIALIZATION

In this module, we initialize the given image (i.e.) get the input image from user by using the keyword 'uigetfile'. This contains only the pathname and filename. To read the image filename, we used 'imread' command. This read image was store in a variable as a matrix. Then we estimate the size of the given image using 'size' command. This given information of size of given image to estimate whether the given text was within the size of input image.

6.2. IMAGE ENCRYPTION

Assume the original image with a size of N1XN2 is in uncompressed format and each pixel with gray value falling into [0, 255] is represented by 8 bits. Denote the bits of a pixel as $b_{i,j,0}, b_{i,j,1}, \ldots, b_{i,j,7}$ where 1<=i<=N1 and1<=j<=N2, the gray value as, and the number of pixels as N(N=N1XN2). That implies

$$b_{i,j,u} = [p_{i,j}/2^u] \bmod 2, \quad u=0,1,2,\ldots,7$$

In encryption phase, the exclusive-or results of the original bits and pseudo-random bits are calculated.

6.3. DATA EMBEDDING

In the data embedding phase, some parameters are embedded into a small number of encrypted pixels, and the LSB of the other encrypted pixels are compressed to create a space for accommodating the additional data and the original data at the positions occupied by the parameters. The detailed procedure is as follows. According to a data-hiding key, the data-hider pseudo-randomly selects N_p encrypted pixels that will be used to carry the parameters for data hiding.

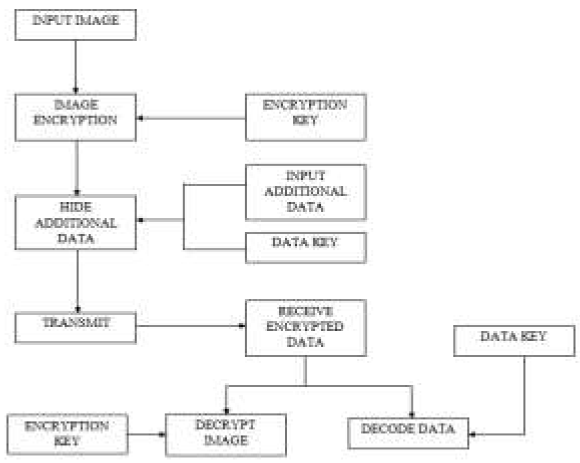


Fig.5.System architecture

6.4 DATA EXTRACTION AND IMAGE RECOVERY

In this phase, we will consider the three cases that a receiver has only the data-hiding key, only the encryption key, and both the data-hiding and encryption keys, respectively.

With an encrypted image containing embedded data, if the receiver has only the data-hiding key, he may first obtain the values of the parameters and from the LSB of the selected encrypted pixels. Note that because of the pseudo-random pixel selection and permutation, any attacker without the data-hiding key cannot obtain the parameter values and the pixel-groups, therefore cannot extract the embedded data. Furthermore, although the receiver having the data-hiding key can successfully extract the embedded data, he cannot get any information about the original image content.

6.5 COMPUTE PSNR VALUE

In this module we compute the PSNR value for input image and decrypted image. Peak Signal-to-Noise Ratio, often abbreviated PSNR, is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale.

PSNR is most easily defined via the mean squared error (*MSE*). Given a noise-free $m \times n$ monochrome image I and its noisy approximation K, *MSE* is defined as:

IJMTARC – VOLUME – VI – ISSUE – 24, OCT-DEC, 2018

ISSN: 2320-1363

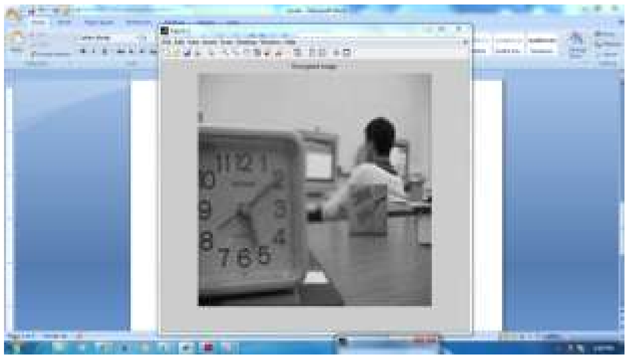
$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

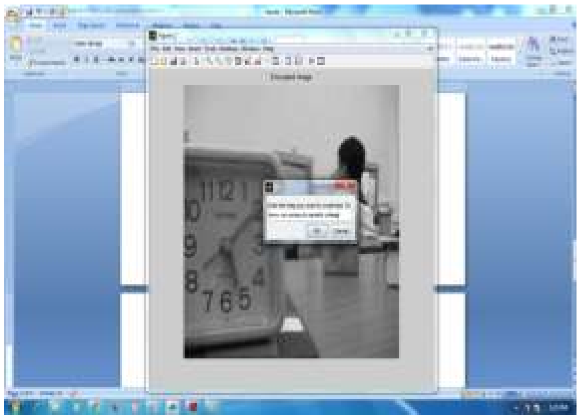
The PSNR is defined as:

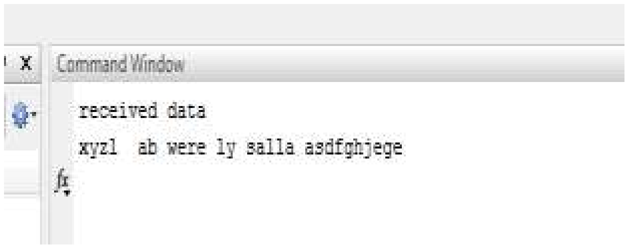
$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \end{aligned}$$

Using this PSNR value we can compare our algorithm with other algorithm that our method gives better result than previous method.

VII.RESULTS

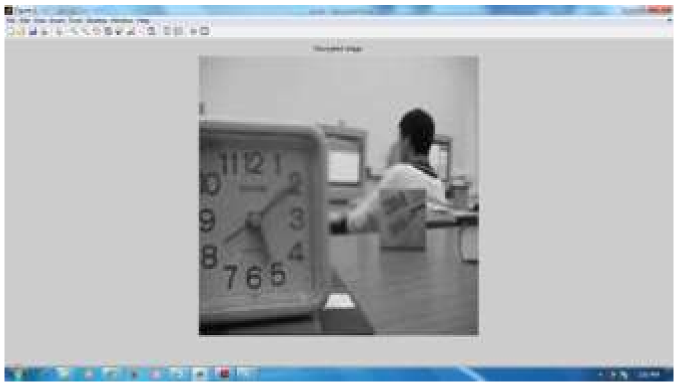






```
received data
xyzl  ab were ly salla asdfghjege
```

```
received data
tmrw cm coming to saradhi college
```



VIII.CONCLUSION

This work proposes a lossless, a reversible, and a combined data hiding schemes for cipher-text images encrypted by public key cryptography with probabilistic and homomorphic properties. In the lossless scheme, the cipher text pixel values are replaced with new values for embedding the additional data into the LSB-planes of cipher text pixels. This way, the embedded data can be directly extracted from the encrypted domain, and the data embedding operation does not affect the decryption of original plaintext image. In the reversible scheme, a preprocessing of histogram shrink is made before encryption, and a half of cipher text pixel values are modified for data embedding. On receiver side, the additional data can be extracted from the plaintext domain, and, although a slight distortion is introduced in decrypted image, the original plaintext image can be recovered without any error. Due to the compatibility of the two schemes, the data embedding operations of the lossless and the reversible schemes can be simultaneously performed

IJMTARC – VOLUME – VI – ISSUE – 24, OCT-DEC, 2018

ISSN: 2320-1363

in an encrypted image. So, the receiver may extract a part of embedded data in the encrypted domain, and extract another part of embedded data and recover the original plaintext image in the plaintext domain.

BIBLIOGRAPHY

[1] N. A. Saleh, H. N. Boghdad, S. I. Shaheen, A. M. Darwish, "High Capacity Lossless Data Embedding Technique for Palette Images Based on Histogram Analysis," Digital Signal Processing, 20, pp. 1629−1636, 2010.

[2] J. Tian, "Reversible Data Embedding Using a Difference Expansion," IEEE Trans. on Circuits and Systems for Video Technology, 13(8), pp. 890−896, 2003.

[3] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su, "Reversible Data Hiding," IEEE Trans. on Circuits and Systems for Video Technology, 16(3), pp. 354−362, 2006.

[4] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Lossless Generalized-LSB Data Embedding," IEEE Trans. on Image Processing,

[5] X. Hu, W. Zhang, X. Li, and N. Yu, "Minimum Rate Prediction and Optimized Histograms Modification for Reversible Data Hiding," IEEE Trans. on Information Forensics and Security, 10(3), pp. 653-664, 2015.

[6] X. Zhang, "Reversible Data Hiding with Optimal Value Transfer," IEEE Trans. on Multimedia, 15(2), 316−325, 2013.

[7] W. Zhang, X. Hu, X. Li, and N. Yu, "Optimal Transition Probability of Reversible Data Hiding for General Distortion Metrics and Its Applications," IEEE Trans. on Image Processing, 24(1), pp. 294-304, 2015.

[8] S. Lian, Z. Liu, Z. Ren, and H. Wang, "Commutative Encryption and Watermarking in Video Compression," IEEE Trans. on Circuits and Systems for Video Technology, 17(6), pp. 774−778, 2007.

[9] M. Cancellaro, F. Battisti, M. Carli, G. Boato, F. G. B. Natale, and A. Neri, "A Commutative Digital Image Watermarking and Encryption Method in the Tree Structured Haar Transform Domain," Signal Processing: Image Communication, 26(1), pp. 1−12, 2011.

[10] X. Zhang, "Commutative Reversible Data Hiding and Encryption," Security and Communication Networks, 6, pp. 1396−1403, 2013.

[11] X. Zhang, "Reversible Data Hiding in Encrypted Image," IEEE Signal Processing Letters, 18(4), pp. 255−258, 2011.

[12] W. Hong, T.-S. Chen, and H.-Y. Wu, "An Improved Reversible Data Hiding in Encrypted Images Using Side Match," IEEE Signal Processing Letters, 19(4), pp. 199−202, 2012.

[13] J. Yu, G. Zhu, X. Li, and J. Yang, "An Improved Algorithm for Reversible Data Hiding in Encrypted Image," Proceeding of the 11th International Workshop on Digital-Forensics Watermark (IWDW 2012), Shanghai, China, Oct. 31-Nov. 02, 2012, Lecture Notes in Computer Science, 7809, pp. 358-367, 2013.

[14] W. Puech, M. Chaumont, and O. Strauss, "A Reversible Data Hiding Method for Encrypted Images," Security, Forensics, Steganography, and Watermarking of Multimedia Contents X, Proc. SPIE, 6819, 2008.

[15] X. Zhang, "Separable Reversible Data Hiding in Encrypted Image," IEEE Trans. Information Forensics & Security, 7(2), pp. 526−532, 2012.

[16] Z. Qian, X. Zhang, and S. Wang, "Reversible Data Hiding in Encrypted JPEG Bitstream," IEEE Trans. on Multimedia, 16(5), pp. 1486−1491, 2014.

[17] M. S. A. Karim, and K. Wong, "Universal Data Embedding in Encrypted Domain," Signal Processing, 94, pp. 174-182, 2014.

[18] K. Ma, W. Zhang, X. Zhao, N. Yu, and F. Li, "Reversible Data Hiding in Encrypted Images by Reserving Room Before Encryption," IEEE Trans. Information Forensics & Security, 8(3), pp. 553-562, 2013.