Covert Image Transfer: A Secret Sharing Technique

Subharag Sarkar
Computer Science Department
W.B.U.T,Kolkata, India

Akash Dasgupta
Computer Science Department
W.B.U.T,Kolkata, India

Aloke Bhakta
Computer Science Department
W.B.U.T,Kolkata, India

ABSTRACT
In cryptography, the art of secret sharing, is to share among a group of selected few an individual share of the secret. Individual shares are of no use on their own. The multifaceted implementation of this scheme in literature including Shamir, Blakley and Asmuth-Bloom to name a few, has led to the emergence of higher computational complexity during both sharing and reconstruction and generates noise like shares. In order to come up with a meaningful sharing scheme, Lin and Tsai proposed a method that uses Steganography using Shamir's secret sharing scheme; again which led to high computational complexity. Therefore, in order to overcome the above problem, a new scheme is suggested which deploys simple graphical masking, done by simply **ANDing**, value substitution for share generation and reconstruction can be done by value re-substitution and then **ORing** the qualified set of shares. Also, this proposed method finally creates, meaningful shares by using Steganography; instead of noise like shares.

General Terms
Cryptography, Secret Sharing

Keywords
Secret Sharing, Steganography, Low computational complexity, High level encryption

1. INTRODUCTION
Steganography, a method employed for the protection of data from illicit attacks, adds to the deceptiveness, and efficacy of the process, hides a secret message in a cover medium. The cover could be a digital image, video, audio, or even html code. Attackers fail to guess whether the cover medium has hidden secret data or not thus providing complete secrecy [1]. However a common weakness of this technique is that an entire protected data or image is kept in a single cover medium. The secret data or image cannot be reconstructed if the medium is lost or corrupted. Secret sharing [2] is the technique that deals with such problem, namely, sharing a highly sensitive secret among a group of **n** users so that only when a sufficient number **k** of them comes together, the secret can be constructed.

The maverick idea of secret sharing was proposed individually by Adi Shamir and George Blakley in 1979 [2]. Shamir's scheme was based on polynomial interpolation whereas Blakley's scheme was based on hyper plane geometry.

The drawback of the aforementioned secret sharing schemes is the requirement of the revelation of the secret during the reconstruction phase [7]. This vulnerability of the system can be effaced if the subject function can be computed without revealing the secret shares or reconstructing the secret back which is known as **function sharing problem** where the function's computation is distributed according to underlying SSS such that distributed parts of computations are carried out by individual user and then the partial results can be combined to yield the final result without disclosing the individual secrets [3][4].

In the year 2004, the method proposed by Lin and Tsai used Steganography for generation of meaningful shares with secret image sharing. The proposal by Yang et al. advocated a method to overcome some of the weaknesses which were apparent in the method of 2007. Nevertheless, both the studies used polynomial based secret sharing approach proposed by Shamir that led to high computational complexity.

In this paper a secret sharing scheme is suggested, different from any of the schemes discussed so far, where simple **ANDing** operation and value substitution is used for share generation and reconstruction can be done by value re-substitution and then simply **ORing** the predefined minimal set of shares. Not only the generated shares are kept into meaningful cover images, but the overtness of the scheme depends upon the technique of generating individual masks to be used for **ANDing** over the original secret for share generation.

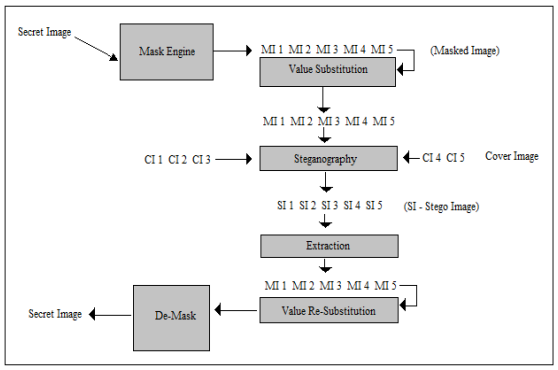

Fig 1: The proposed Plan

2. PROPOSED METHOD
The proposed work is based upon a new, simple, innovative secret sharing scheme. The procedure for creating the **n** secret shares can be divided into the following phases:

2.1. Encoding
2.1.1. Mask Generation and Masking
Let's consider any secret as binary bit file. The bit file of any size will be decomposed into **n** shares in such a way that the original file can be reconstructed using **ORing** any **k** number of shares where **k<=n>=2** but in practice it should be considered as **2<=k<n>=3**.

The basic concept of this method is that every share should have some bits missing and those bits should be replenished by exactly (**k-1**) other shares but not less than that [14][15]. Thus the mask created follows the above rules and should be repeatedly **ANDed** over the secret in any regular order.

Different masks will produce different shares (The style of placing the mask over the secret could be anything but it will be same for every share. It may also be noted that the knowledge of positioning the mask over the secret is not at all required for reconstruction of the secret) from the original secret. [5][6].Thus 0 on the mask will eliminate a particular bit of the secret and 1 will retain the bit to form a share [8].

On **ORing** any **k** number of shares, the secret can be recreated but random number of 0's and 1's reflect no idea of the secret. As an example a possible set of masks for 5 shares with threshold of 3 shares is shown.

Share 1: 1 1 1 1 1 1 0 0

Share 2: 1 1 1 0 0 0 1 1

Share 3: 1 0 0 1 1 0 1 1

Share 4: 0 1 0 1 0 1 1 0

Share 5: 0 0 1 0 1 1 0 1

One can easily check that **ORing** any three or more shares gives us all 1's but with less than three shares, some position still have 0's i.e. missing bits.

The pseudo code of this phase is as follows:

Algorithm: Generating masks and using them to create masked images.

Input: Secret Image

Output: Masked Images ready for next phase

Steps:

1. Create a 2d array mask[5][8] to store the new masks
2. Initialize srand(time(NULL)) which helps to create masks
3. for j=0 to 7
   Let c=0,d=0,i=0
   while i<5

k=rand()%2//Extracts 0 or 1 randomly

If k==1 && c<3 //If k=1, and number of 1s in column is less than 3, it stores 1 and increases c by 1

mask[i][j]=k

c++

Else

continue

Else if k==0 && d<2// If k=0, and number of 0s in column is less than 2, it stores 0 and increases d by 1.

mask[i][j]=k
d++

Else

Continue

End IF

End IF.

i++//the row number increases by one.

End While
4. Repeat Step 3 as long as the for loop ends.
5. Extract a value from secret file and store in k
6. Convert value of k in its binary counterpart
7. Multiply each bit with each element of each row of mask[][] to get the five masked values and store them in arrays m1[],m2[],m3[],m4[],m5.
8. Repeat step 5, 6 and 7 until all values are converted and the masked images are created.

2.1.2. Value Substitution:

This phase provides an extra level of encryption over the masked shares. The masked share has gone through controlled destruction of bytes, so each share cannot reveal the original secret. In this step the bytes of data are substituted using a special **List** to generate noise like shares which is then sent for steganography [9][10][11].

The value substitution method is inspired from the **monoalphabetic** cipher technique. In **monoalphabetic** cipher, each alphabet has a possible $4x10^{26}$ keys, similarly in this method each byte of data has a possible $4x10^{255}$ keys. This makes the cipher harder to decrypt. So even if the time complexity increases a bit, the high level of encryption provides more security to the data.

The pseudo code of this phase is as follows:

Algorithm: Adding another layer of encryption to make the secret more secure.

Input: Masked Images of previous phase, **List** for encryption.

Output: Encrypted masked images ready for next phase.

Steps:

1. Initialize a masked image in m[0…..n] and array dec[256] contains the values of the **List**.
2. For i=0 to n
   m=dec[m[i]]// the value at m[i] acts as index to produce the substituted value for the new masked image
   End Loop
3. Repeat 1 and 2 for all the masked images to create the new masked images.

2.1.3. Steganography

In this phase we use Steganography to hide the five shares in five cover images. The proposed method has removed the problem in Steganography (Hiding whole image in one cover image) [12][13]. This phase hides the noise like shares using LSB technique to avoid suspicion and create a safe way to convey the secret.

The pseudo code of this phase is as follows:

Algorithm: Hide the five masked shares inside five cover images.

Input: Masked Shares M1, M2, M3, M4, M5 and Cover Images C1, C2, C3, C4, C5

Output: Stego Images S1, S2, S3, S4, S5

Steps:

1. Start a loop from 0 to n to traverse through all the values.
2. Extract the value of a particular index from masked image and convert it into its binary counterpart.

3. Break the binary number into three parts, k1 contains the first three bits, k2 contains the middle three bits and k3 contains the last two bits.
4. Extract the pixel of that index of the cover image and convert it into binary.
5. Insert k1 in the last three bits of the red value,
6. Insert k2 in the last three bits of the green value,
7. Insert k3 in the last two bits of the blue value.
8. The red, green and blue values are converted to their final form and stored in the final image.
9. Repeat Step 2 to Step 8 to hide the masked image inside the cover image.
10. Repeat Step 1 to Step 9 for all the five masked images, to create the five Stego Images.

The Stego Images are now ready to be sent to the sender.

2.2. Decoding

2.2.1. Retrieval from Cover Image

The retrieval process is easy and takes very little time. Tracing back the steps used earlier, the masked images are extracted from the cover images.

The pseudo code of this phase is as follows:

Algorithm: Extracting masked image from cover image

Input: Stego Images S1, S2, S3, S4, S5

Output: Masked Images M1, M2, M3, M4, M5

Steps:

1. Start a loop from 0 to n to traverse through the image.
2. Extract the pixel of a particular index of the cover image.
3. Convert the red, green and blue values into their binary counterparts.
4. Extract last three bits of red value and store in k1
5. Extract last three bits of green value and store in k2
6. Extract last three bits of blue value and store in k3
7. Join k1, k2, k3 to get the hidden byte.
8. Convert the value into decimal and store it in array m[].
9. Repeat Step 2 to Step 8 for all the pixels
10. Repeat Step 1 to Step 9 for the five stego images to retrieve the five masked image.

2.2.2. Value Re-Substitution

After the masked images have been retrieved, the values are re-substituted back to the original values of the masked shares using the same **List** that is used for mono alphabetic substitution.

The pseudo code of this phase is as follows:

Algorithm: Re-substitute the original values of the masked images

Input: Masked Images M1, M2, M3, M4, M5 and **List** for decryption

Output: Original Masked Images M1, M2, M3, M4, M5

Steps:

1. Initialize m1[0….n] to store the masked image and dec[256] to store the content of **List**
2. For i=0 to n-1
   For j=0 to 255// If the values matched, the index of dec shows the original value of the masked image
   If m[i] ==dec[j]
   m[i]=j
   End IF
   End Loop
   End Loop
3. Repeat Step 1 and Step 2 to get the original masked images

2.2.3. De-Masking

After value re-substitution, the original masked images are retrieved. To find the original image we require at least three (k) masked images if five (n) are created. A value of a particular position is taken from every masked image, converted to binary and the bits are ORed to get the original data back. After the whole operation is completed the original secret data is ready for use [18].

The pseudo code of this phase is as follows:

Algorithm: Recreating the original Secret Image

Input: Masked Images M1, M2, M3

Output: Secret Image S

Steps:

1. Start a loop from 0 to n to traverse through all the values.
2. For a particular index j store the values of the three images in a, b, c.
3. Convert the values into binary counterpart and store them into three arrays x[], y[], z[].
4. For i=0 to 7
   If x[i]+y[i]+z[i]>0// If any of the bits is 1, the original bit is 1 else it is 0.
   K[i]=1
   Else
   K[i]=0
   End IF
   End Loop
5. Repeat Step 4 until loop ends.
6. Convert the value in k[] into decimal and store I in S[j].
7. Repeat Step 2 to Step 6 until all the values are recreated.

3. EXPERIMENTAL RESULTS

To check the usefulness of the proposed scheme, the values of PSNR is used which indicates the success or failure of the method. PSNR or Peak Signal to Noise Ratio is used to check the difference between the same image before and after the change. Higher value of PSNR indicates that no changes can be seen by the naked eye.

The following images are used as Secret image and Cover images.

Fig 2: Secret Image

Fig 3: Cover Image 1

Fig 4: Cover Image 2

Fig 5: Cover Image 3

Fig 6: Cover Image 4

Fig 7: Cover Image 5

The PSNR values found after the Steganography process is:

Cover Image 1: 39.679821 db
Cover Image 2: 38.789654 db
Cover Image 3: 41.256935 db
Cover Image 4: 40.586789 db
Cover Image 5: 37.458962 db

And the PSNR value of the secret image after recreation is 99.256958 db. These PSNR values indicate that no considerable changes can be seen by the naked eye.

4. CONCLUSION

In this paper, a new secret sharing approach has been proposed with acceptable computational overhead if not nil. This is one of the simplest and efficient thresholds sharing scheme, practically having acceptable computational overhead during both share generation and secret reconstruction. The previous methods have some problems which has been addressed by this method and an extra layer of encryption guarantees more safety to the secret.

5. REFERENCES

[1] Shamir, Adi (1979). "How to share a secret".Communications of the ACM 22 (11): 612–613.

[2] Blakley, G. R. (1979). "Safeguarding cryptographic keys". Proceedings of the National Computer Conference 48: 313–317.

[3] Krawczyk, Hugo (1993). Secret Sharing Made Short (PDF). CRYPTO '93.

[4] Rabin, Michael O. (1989). "Efficient dispersal of information for security, load balancing, and fault tolerance". Journal of the ACM 36 (2).

[5] Resch, Jason; Plank, James (February 15, 2011).

[6] Parakh, A. and Kak, S. Space efficient secret sharing for implicit data security. Information Sciences, vol. 181, pp. 335-341, 2011.

[7] Parakh, A. and Kak, S. Online data storage using implicit security. Information Sciences, vol. 179, pp. 3323-3331, 2009.

[8] ‘"Unvanish: Reconstructing Self-Destructing Data'

[9] Liddell, Henry George; Scott, Robert; Jones, Henry Stuart; McKenzie, Roderick (1984). A Greek-English Lexicon. Oxford University Press.

[10] Rivest, Ronald L. (1990). "Cryptology". In J. Van Leeuwen. Handbook of Theoretical Computer Science 1. Elsevier.

[11] Bellare, Mihir; Rogaway, Phillip (21 September 2005). "Introduction". Introduction to Modern Cryptography. p. 10.

[12] Menezes, A. J.; van Oorschot, P. C.; Vanstone, S. A. Handbook of Applied Cryptography. ISBN 0-8493-8523-7.

[13] "Overview per country". Crypto Law Survey. February 2013. Retrieved 26 March2015.

[14] "UK Data Encryption Disclosure Law Takes Effect". PC World. 1 October 2007. Retrieved 26 March 2015.

[15] Doctorow, Cory (2 May 2007). "Digg users revolt over AACS key". Boing Boing. Retrieved 26 March 2015.

[16] Kahn, David (1967). The Codebreakers. ISBN 0-684-83130-9.

[17] Oded Goldreich, Foundations of Cryptography, Volume 1: Basic Tools, Cambridge University Press, 2001, ISBN 0-521-79172-3

[18] "Cryptology (definition)". Merriam-Webster's Collegiate Dictionary (11th ed.). Merriam-Webster. Retrieved 26 March 2015.