ON THE SEMANTIC SECURITY OF SECRET IMAGE SHARING METHODS

Shreelatha Bhadravati, Pradeep K Atrey and Majid Khabbazian

University of Winnipeg and University of Alberta

ABSTRACT

In this work, we analyze some of the existing secret image sharing methods and show that they do not possess indistinguishability, a property of many secure systems. We propose a new method based on the (k, n) threshold secret sharing scheme for images in the compressed and uncompressed domains. Our method generates minimum share sizes with similar computational cost as the previous methods, yet it is computationally secure and satisfies the indistinguishability property.

Index Terms— Secret image sharing, cryptanalysis

1. INTRODUCTION

The task of securing digital images has been studied extensively in the past decade. The conventional approach to protect secret images is to use block encryption methods such as the Advanced Encryption Standard (AES). The block encryption techniques requires a key, which increases vulnerability to attacks. Keeping the key at a single place or with a single person is vulnerable to single point failure, as one person cannot be trusted with the security of the key. Storing the key at multiple locations further increases the possibility of an attack because it increases the chances of access to more adversaries.

In 1979, Shamir developed the first secret sharing method [1]. The basic idea of his (k, n) threshold method is that the secret to be shared is distributed to n participants and any $k \leq n$ participants together can reconstruct the secret. Any number of participants less than k cannot reconstruct the secret. Applying Shamir secret sharing on images provides information theoretically secure image sharing. However, if the size of shares is important this basic scheme may not be preferable because, using Shamir's secret sharing, the size of shares would be the same as the size of the secret image. Ideally, we prefer the size of shares to be $\frac{1}{k}$ of that of the size of the secret image. Unfortunately, it is well known that we cannot reduce the size of shares if we wish to have information theoretic secrecy instead of computational secrecy [2].

Recently, many methods have been proposed to reduce the size of shares in the *computationally secure model* [3] [4]. Although it is not mentioned in their work, their main idea is to use the Reed-Solomon error correction scheme, which has similarity to Shamir's secret sharing scheme. Reed-Solomon correction scheme can be used for information dispersal but not for hiding information as we will show in this paper.

In computationally secure model, a basic method is, perhaps, to encrypt the image using a secret key and then apply Shamir's scheme only on the secret encryption key. In this method, every user will get the same copy of the encrypted image and a share of the secret encryption key. Although this method may be computationally more efficient (depending on k, n and the encryption method) than the basic method mentioned earlier, the size of shares is again the same as the size of the original image. In [2], it was proposed to use the above method followed by an information dispersal scheme to reduce the size of shares. Our proposed method is based on this general approach. In particular, our proposed method mixes a simple stream cipher with the Reed-Solomon correction method to reduce the computational complexity and to reduce the size of shares to $\frac{1}{k}$ of that of the original image.

2. CRYPTANALYSIS OF EXISTING SECRET IMAGE SHARING SCHEMES

In this section, we first discuss the implementation of the existing secret image sharing schemes. Then, we perform cryptanalysis to show that they are semantically insecure. All the mathematical operations throughout this paper are in finite fields.

2.1. Thein and Lin's Method

Thein and Lin [3] proposed a secret image sharing scheme based on the Reed-Solomon correction scheme. In their proposed method, the secret image is divided into several sections in such a way that each section has k pixels and each pixel of the image belongs to only one section. The polynomial of the order $(k-1)$ is constructed using the k consecutive pixels of the same section, as follows:

$$f_j(x) = \sum_{i=0}^{k-1} d_{i,j} \times x^i \tag{1}$$

where $f_j(x)$ is the polynomial function for the j^{th} section, $d_{i,j}$ is the value of the i^{th} pixel in the j^{th} section. The m^{th} share image is then the pair (m, s_m), where $s_m =$

$(f_1(m), f_2(m), \dots)$. In this method, the size of the share images is reduced by $\frac{1}{k}$ as the number of sections is $\frac{1}{k}$ of the size of the image.

2.2. Alharthi and Atrey's Method

Alharthi and Atrey [4] proposed an improvement to Thein and Lin's [3] method. In this method, the secret image is divided into k sections, and the polynomial is constructed using the pixel values from each of these k sections as follows:

$$f_j(x) = \sum_{i=0}^{k-1} d_{j,i} \times x^i \tag{2}$$

where $d_{j,i}$ is the value of the j^{th} pixel in the i^{th} section. The m^{th} share image is again (m, s_m), where $s_m = (f_1(m), f_2(m), \dots)$. Note that in this method the coefficients of a polynomial come from different sections while in the Thein and Lin's method they come from a single section.

2.3. Lin and Tsai's Method

Lin and Tsai [5] proposed a (k, n) secret sharing scheme for compressed images. Using the first coefficient value in the DCT block as a seed, a sequence of random numbers are generated to replace the initial coefficients in the 8×8 block. The 2^{nd} through 10^{th} coefficients are calculated as $C'_m = R_m \times C_m$ respectively, where C'_m is the new value generated, R_m is the random number generated in the range $[0, C_1]$, C_1 is the first coefficient value, C_m is the original AC coefficient value, and m is $2, 3, \dots, 10$. The remaining AC coefficients in the DCT block are discarded. Then the first coefficient C_1 is encrypted using Shamir's secret sharing scheme into n shares and p is taken as the nearest prime number larger than C_1. Random numbers are generated in the range $[0, p]$.

2.4. Security Issue of Existing Schemes

Let I be an image which we represent as a sequence of numbers, i.e., $I = (a_1, \dots, a_r)$, where $(a_1, \dots, a_r)$ are the pixel values of the image I and r is the total number of pixels in I. For simplicity, we assume that k divides r, where k is the minimum number of users required to generate the original image for (k, n) secret image sharing. For security reasons, in Thein and Lin's [3] method, a random permutation is applied to I. The random permutation is computed using a permutation key K. Thein and Lin [3] proposed keeping the permutation key with the owner or shared among the users. To analyze the security of their method, we consider two cases: i) attacker has access to the permutation key, ii) attacker does not possess the permutation key. In both cases, we show that their method and its variants are insecure.

2.4.1. Case 1: Attacker has access to the permutation key

In the first case, suppose that the attacker has the following: i) the permutation key K, ii) a single share image (m, s_m), and iii) an image J. Having a single share, the objective of the attacker is to verify whether J is the secret image. Clearly, the secret sharing scheme is not secure if an attacker can do the above verification with a non-negligible probability.

To do the verification, the attacker can use the following basic verification method:

1. Apply the secret sharing method on J. Note that this is possible since attacker has the permutation key.
2. Compare the generated share image with s_m.
3. Return *true* if there is a match; return *false* otherwise.

Note that if J is the secret image, the above basic verification method will always return true. The following proposition states that the probability of false alarm is very small if J is a random image. Note that the same method can be used against Alharthi and Atrey [4] method; the difference being that the values used to construct the polynomial equation will be the first unused pixels of each of k sections.

Proposition 1. *Let I be the secret image, (m, s_m) be a share generated by the Thein and Lin's method, and J be a random image (i.e., J is a sequence of independent and identically distributed (i.i.d) random numbers with uniform distribution). The probability that the verification method apply to J returns true is $\exp(-\frac{r}{k} \log q)$, where q is the size of the finite field.*

Proof sketch. Let s'_m be share generated by applying the secret sharing method on J. Since J is a random image with uniform distribution, s'_m will also be a sequence of i.i.d random numbers (from the finite filed) with uniform distribution. Therefore, we have

$$Pr(s'_m = s_m) = Pr\left(\bigwedge_{i=1}^{\frac{r}{k}} (s'_m(i) = s_m(i))\right) = \left(\frac{1}{q}\right)^{\frac{r}{k}},$$

where $s_m(i)$ ($s'_m(i)$) denotes the i^{th} number in s_m (s'_m). □

2.4.2. Case 2: Attacker does not have the permutation key

In the second case, we assume that the attacker does not possess the permutation key K. We show that the attacker can still gain information by having only one share. Suppose that the attacker has the share $(1, s_1)$. We have

$$\sum_{i=0}^{r/k} s_1(i) = \sum_{a \in I} a.$$

Therefore, an image J cannot be the secret image if

$$\sum_{i=0}^{r/k} s_1(i) \neq \sum_{a \in J} a \tag{3}$$

An attacker can verify whether J is the secret image by using (3), for which it does not require to know the permutation key. In this case the probability of false alarm is $\frac{1}{q}$ if J is a random image.

Proposition 2. *Let I be the secret image, $(1, s_1)$ be a share generated by the Thein and Lin's method, and J be a random image (i.e., J is a sequence of i.i.d random numbers with uniform distribution). Then*

$$Pr\left(\sum_{i=0}^{r/k} s_1(i) = \sum_{a \in J} a\right) = \frac{1}{q}.$$

Proof sketch. Note that $\sum_{a \in J} a$ is a random finite field number with uniform distribution. □

We can extend this attack to Alharthi and Atrey's [4] method. Similarly, we can prove that Lin and Tsai's [5] method is semantically insecure. We know that the adversary has a share image (m, s_m). The adversary selects an image J which is to be verified and obtains its 64 DCT coefficients of 8×8 block. Now the adversary can pick the first value of each block, and use it as a seed to generate a sequence of random numbers. Then it can be compared to the AC values of the share image since these are made public. If the values of the share image does not match the generated value C'_I of image J then we know that the image J is not the secret image.

3. PROPOSED METHOD

To overcome the above-mentioned weakness of the existing secret image sharing methods, we propose a simple and efficient (k, n) threshold scheme. We show the implementation of the proposed scheme for uncompressed and compressed images. The two phases (share generation and secret reconstruction) of our scheme are described as follows.

3.1. Share Generation Phase

1. Perform encryption on the secret image I. In uncompressed domain, the pixels values of the image and in compressed domain the DCT coefficients are added with pseudorandom numbers generated using a cryptographically secure pseudo random number generator to obtain the encrypted image E.
2. Using Reed Solomon error correction, the encrypted image E is partitioned into n fragments $E_1, E_2, \ldots, E_n$.
3. Using Shamir's secret sharing, we generate n shares of the seed $(K_1, K_2, \ldots, K_n)$, used to generate random numbers for the encryption part.
4. Each share $s_i = (E_i, K_i)$, $1 \leq i \leq n$, is distributed to the i^{th} user.

Fig. 1. Color image of Lena and its first three shares

Note that the security of the above scheme depends on the security of the pseudo random function used.

3.2. Secret Reconstruction Phase

1. Collect any k shares from the participants.
2. Using Reed-Solomon correction scheme reconstruct E from the shares collected, $E_i, i = 1, 2, \ldots, k$.
3. Using Shamir's secret sharing scheme recover the seed value K out of $K_i, i = 1, 2, \ldots, k$ shares.
4. Decrypt E to recover the secret image I, by first generating random numbers using the seed constructed in the previous step, and then subtracting the random numbers from the reconstructed encrypted image E.

4. RESULTS AND ANALYSIS

4.1. Experimental Results

We have used Lena and Baboon (gray scale and color) images for our experiments. Figure 1 and Figure 2 show the share images generated for Lena color and gray Baboon images, respectively. We compare the methods for computational speed in both compressed and uncompressed domains experimentally (by reporting the actual computation times) as well as analytically (based on the number of operations performed in each method and the number of random values used for share creation). The share creation times for various uncompressed and compressed images are shown in Table 1 and Table 2, respectively. These times were recorded using a i5 processor, 2.67GHz Intel machine with 4 GB RAM. As per the results it can be seen that our method is a little faster when compared to [3] [5] and slightly slower than [4] as shown in Table 1 and Table 2,and perceptually secure as shown in Figure 1 and Figure 2 where the share images are noisy and reveal no information about the secret image and there is a reduction in the size of the share image by $\frac{1}{k}$ similar to [3], [4].

Considering the number of finite field operations required in each method, Thein and Lin's method has to perform a permutation of r pixels, $(k-1) \times \frac{r}{k}$ addition and $(k-1) \times \frac{r}{k}$

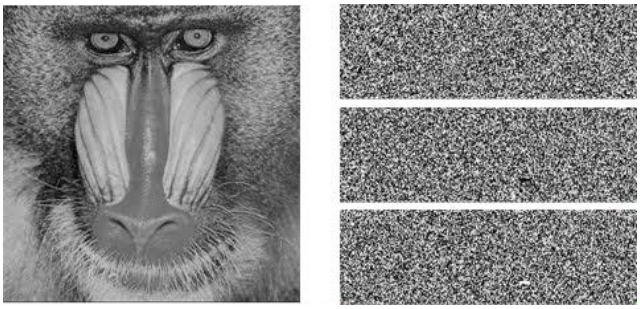

Fig. 2. Baboon gray image and its first three shares

Table 1. Share creation times for uncompressed images

Image	Thein and Lin [3]	Alharthi and Atrey [4]	Proposed method
Lena-gray	147ms	94ms	96ms
Baboon-gray	133ms	83ms	90ms
Lena color	177ms	101ms	109ms
Baboon color	158ms	80ms	97ms

multiplication operations. Although Alharthi and Atrey's [4] method has to perform only $(k-1) \times \frac{r}{k}$ additions and $(k-1) \times \frac{r}{k}$ multiplications, it involves generating new values of x everytime equation (2) is used. Our proposed method has to perform $(2k-1) \times \frac{r}{k}$ additions and $(k-1) \times \frac{r}{k}$ multiplication operations in uncompressed domain. In compressed domain, Lin and Tsai's [5] method performs $(k-1) \times \frac{r}{8\times 8}$ addition and $(10 \times \frac{r}{8\times 8} + 2 \times \frac{r}{8\times 8})$ multiplications. Our proposed method has to perform $(2k-1) \times \frac{r}{8\times k}$ additions and $(k-1) \times \frac{r}{8\times k}$ multiplication operations. Our method has slightly more computational overhead than [4] and has comparable computational overhead with [3] and [5] even though our method has more number of addition operations to be performed, since [3] implements a permutation step which is not performed in our method and the number of multiplication operations to be performed in [5] is very much higher when compared to our method, which conforms to the experimental results (Table 1 and Table 2). Furthermore, in terms of the number of random bits required,Thein and Lin's [3] method needs $\theta(r \log r)$ random bits to perform the random permutation while our method requires $\theta(r \log q)$ random bits for the stream cipher, where q is the size of the finite field used. Note that q is typically less than r in practice. Hence we show that our method has the same computational cost as the other methods based on the number of finite field operations performed.

4.2. Security Analysis

Suppose that the attacker has access to $k-1$ shares, and wishes to know whether a given image J is the secret image. Following we informally explain why the attacker is computationally unable to do this despite of possessing $k-1$ shares. We refer the reader to [2] for more details.

We can assume that the attacker possesses E, as this does not put the attacker in a weaker position. Note that having E, one can generate all the shares, therefore, an attacker with $k-1$ shares is not more powerful than the one with E. If the generated random numbers were truly random, the attacker could not gain any information by possessing E. In other words, for any image J we have $Pr(J = I|E) = Pr(J = I)$. Since it is computationally difficult to distinguish between a sequence of numbers generated by the random number generator and a sequence of true random numbers, having E, the attacker is computationally unable to gain any information.

Table 2. Share creation times for compressed images

Image	Lin and Tsai [5]	Proposed method
Lena	94ms	77ms
Baboon	110ms	83ms

5. CONCLUSIONS

In this paper, we presented a cryptanalysis of some of the existing secret image sharing methods. We showed how to solve the issue by proposing a simple and efficient method that uses random numbers along with the pixel/coefficient value to generate the shares. The proposed method reduces the size of shares to its optimum, that is $\frac{1}{k}$ of the size of the original image. We compared the computation costs of our method with the existing ones through quoting the number of finite field operations and execution time. The results show that our method has similar computation costs as the methods studied in this work.

6. REFERENCES

[1] Adi Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[2] Hugo Krawczyk, "Secret sharing made short," in *Advances in CryptologyCRYPTO93*. Springer, 1994, pp. 136–146.

[3] Chih-Ching Thien and Ja-Chen Lin, "Secret image sharing," *Computers and Graphics*, vol. 26, no. 5, pp. 765 – 770, 2002.

[4] Saeed S. Alharthi and Pradeep K. Atrey, "Further improvements on secret image sharing scheme," in *ACM International Workshop on Multimedia in Forensics, Security and Intelligence*, Firenze, Italy, 2010, MiFor '10, pp. 53–58.

[5] Chang-Chou Lin and Wen-Hsiang Tsai, "Secret image sharing with capability of share data reduction," *Optical Engineering*, vol. 42, pp. 2340–2345, 2003.