A Key Based Secure Threshold Cryptography for Secret Image

Prabir Kr. Naskar[1], Hari Narayan Khan[2], Atal Chaudhuri[3]
(Corresponding author: Prabir Kr. Naskar)

Department of Computer Science & Engineering, MCKV Institute of Engineering[1]
Liluah, Howrah-711204, West Bengal, India.
Department of Computer Science & Engineering, Regent Education and Research Foundation[2]
Barrackpore, Kolkata-700121, West Bengal, India.
Department of Computer Science & Engineering, Jadavpur University[3]
Jadavpur, Kolkata-700032, West Bengal, India.
(Email: cse.prabir@gmail.com, manik1984@gmail.com, atalc23@gmail.com)

(Received June 12, 2013; revised and accepted Jan, 22 & Apr. 29, 2014)

Abstract

This paper presents a key based secured (k, n) threshold cryptography where key is used to encrypt the secret and then the secret as well as key is shared among set of n participants. In sharing phase, each secret byte is selected randomly from secret fields depending upon the key. That provides additional protection of the secret data. Also, each share has some bytes missing and these missing bytes can be recovered from a set of exactly k shares. Thus a given byte position can be confirmed for any k shares, but not less than k. Hence k shares are required to give back the secret. As a result, the generated shares are compressed and if k is closer to n then the compression ratio is increased. That provides strong protection of secret data. At the reconstruction phase only when a qualified set of legitimate shares comes together then reconstruction is possible. The proposed scheme is described in detail along with its security analysis, such as key sensitivity analysis and statistical analysis. This scheme has been tested using different images to prove that the scheme has great potential and has a good ability to achieve the confidential security.

Keywords: Compression, image Sharing, key-based threshold cryptography, perfect secret sharing (PSS), random selection of secret bytes, statistical analysis

1 Introduction

Protection of sensitive data is an important issue, during transmission over internet. Many cryptographic techniques are there to protect secret data. However, a common weakness of these technique is that an entire secret data is kept in a single medium. The secret data cannot be revealed if the medium or key is lost or corrupted. This is termed as a single point failure. To overcome this drawback secret sharing becomes more popular. In the secret sharing scheme, there is one dealer and n participants. The dealer gives a secret to the participants, but only when specific conditions are fulfilled. The dealer accomplishes this by giving each participant a share in such a way that any group of k or more participants (i.e., qualified participant) reconstruct the secret but no group of less than k players can. Such a system is called a (k, n)-threshold based secret sharing scheme. Threshold Secret sharing scheme thus says that: A secret is some data S. Our goal is to divide S into n shares V_1, V_2. $\cdots$, V_n in such a way that:

1) Knowledge of any k or more V_i shares makes S easily computable, where $1 \leq i \leq n$ and $2 < k \leq n$.

2) Knowledge of any $k - 1$ or fewer V_i shares leaves S completely undetermined (in the sense that all its possible values are equally likely).

If $k = n$, then all the shares are required in the (n, n)-threshold scheme to recover the secret. However, the lost of any of the share produced using the (n, n)-threshold scheme results in inaccessible secret messages. Figure 1 shows the conceptual view of a (k, n)-threshold sharing scheme.

Well known secret sharing schemes (SSS) in the literature include Shamir's SSS [16] based on polynomial interpolation, Blakley's SSS [2] based on hyper plane geometry, Asmuth-Bloom's SSS [1] based on Chinese Remainder theorem. Karnin et al. [11] suggested the concept of perfect secret sharing (PSS) where zero information of the secret is revealed for an unqualified group of $(k - 1)$ or fewer members. Specifically, Karnin et al. [11] used a term referred as information entropy (a measurement of the uncertainty of the secret), denoted as $H(s)$ where s is

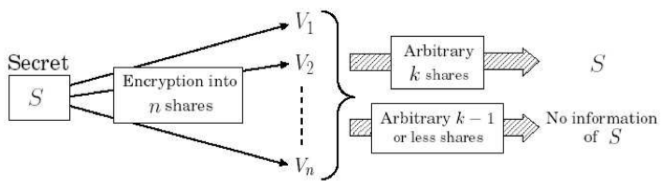


Figure 1: Concept of shared cryptography

a secret shared among n participants. The claim of PSS (Perfect Secret Sharing) schemes must satisfy the following:

1) A qualified coalition of n or more participants, C can reconstruct the secret (s), s: $H(s|C) = 0, \forall|C| \geq k$;

2) An unqualified coalition of $(n-1)$ or few participants, C has no information about the secret (s), s: $H(s|C) = H(s), \forall|C| < k$.

For these requirements in PSS schemes, a secret has zero uncertainty if the secret can be discovered by n or more participants. On the contrary, the secret, in PSS schemes, remains the same uncertainty for $(k-1)$ or fewer members. Therefore, there is no information exposed to the $(k-1)$ or fewer members.

A shortcoming of above secret sharing schemes is the need to reveal the secret shares during the reconstruction phase. The system would be more secure if the subject function can be computed without revealing the secret shares or reconstructing the secret back. This is known as function sharing problem where the function computation is distributed according to underlying SSS such that distributed parts of computation are carried out by individual user and then the partial results can be combined to yield the final result without disclosing the individual secrets. Various function sharing protocols are been proposed [4, 5, 6, 7, 10, 15, 17] mostly based on Shamir secret sharing as the underlying scheme. A better image secret sharing approach was also proposed by Thien & Lin [18]. With some cryptographic computation, they cleverly used Shamir SSS to share a secret image. Chao et al. [3] proposed a method to extend (n, n) scheme to (k, n) scheme by using shadows-assignment matrix. Dong and Ku [8] proposed a new (n, n) secret image sharing scheme with no pixel expansion. In their scheme reconstruction is based on addition which has low computational complexity. Dong et al. [9] proposed a $(2, n)$ secret sharing scheme based on Boolean operation. The reconstructed image is totally the same with the original secret image and the scheme has no pixel expansion and contrast value was ideal. Apart from above secret sharing schemes, we propose a secret sharing scheme, where each share contains partial secret information. As a result, each generated shares are compressed. That provides strong protection of the secret data. In our scheme, each share contains secret data and header data as shared form. A header structure is constructed by the key, k, n and total number of bytes in secret and individual share number. At the reconstruction phase, only when k numbers of shares come together, then original header is reconstructed that is used to reconstruct the original secret. In our scheme secret reconstruction is not possible for less than threshold (k) number of shares; so it is Perfect Secret Sharing scheme.

2 Background and Related Work

2.1 Shamir's Secret Sharing Scheme

Shamir's secret sharing scheme [16] is based on (k, n)-threshold based secret sharing technique ($k \leq n$). The technique allows any k out of n shares to construct a given secret, a $(k-1)$ degree polynomial is necessary. This polynomial function of order $(k-1)$ is constructed as,

$$F(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{k-1} x^{k-1} \bmod p.$$

Now we can easily generate n number of shares by using above equation. Where a0 is the secret, p is a prime number and all other coefficients are random elements from the secret. Each of the n shares is a pair (x_i, y_i) of numbers satisfying $f(x_i) = y_i$ and $x_i > 0$, $1 \leq i \leq$n and $0 < x_1 < x_2 < ... < x_k \leq p-1$. Given any k shares, the polynomial is uniquely determined and hence the secret a0 can be computed via Lagrange interpolation. However, given $k-1$ or fewer shares, the secret can be any element in the field.

The polynomial function f(x) is destroyed after each shareholder possesses a pair of values (x_i, y_i) so that no single shareholder knows the secret value a_0. In fact, no groups of k-1 or fewer shares can discover the secret a_0. On the other hand, when k or more secret shares are available, then we may set at least k linear equations $y_i = f(x_i)$ for the unknown a_i. The unique solution to these equations shows that the secret value a_0 can be easily obtained by using Lagrange interpolation.

2.2 Blakley's Secret Sharing Scheme

Blakley's [2] scheme is less space-efficient than Shamir's, while Shamir's shares are individually as large as the original secret. This scheme uses hyperplane geometry to solve the secret sharing problem. The secret is a point in a k-dimensional space and n shares are affine hyperplanes that pass through this point. An affine hyperplane in a k-dimensional space with coordinates in a field can be described by a linear equation of the following form:

$$a_1 x_1 + a_2 x_2 + a_3 x_3 + ... + a_k x_k = b.$$

The intersection point is obtained by finding the intersection of any k of these hyperplanes. The secret can be any of the coordinates of the intersection point or any function of the coordinates. We take the secret to be the first coordinate of the point of intersection.

2.3 Asmuth-Bloom's Secret Sharing Scheme

A fundamentally different Secret sharing scheme is Asmuth-Bloom's secret sharing scheme [1] which shares a secret among the parties using modular arithmetic and reconstructs it by Chinese Remainder Theorem (CRT). In Asmuth-Bloom's Secret Sharing Scheme, the sharing and Reconstruction of the secret can be done as follows.

Sharing Phase: To share the secret d among a group of n users, the dealer does the following:

1) A set of relatively prime integers $m_0 < m_1 < m_2 < ... < m_n$ where $m_0 > d$ is a prime, are chosen such that $\prod_{i=1}^{k} m_i > m_0 \quad \prod_{i=1}^{k-1} m_{n-i+1}$;

2) Let M denote $\prod_{i=1}^{k} m_i$, the dealer computes,

$$y = d + am_0,$$

where a is a positive integer generated randomly subjected to the condition that $0 \leq y < M$.

3) The share of the i^{th} user $1 \leq i \leq n$, $y_i = y$ mod m_i.

Reconstruction Phase: Assume S is a coalition of k users to reconstruct the secret, let M_s denote$\prod_{i \in S^{m_i}}$.

1) Given the system $y = y_i$ mod m_i. For i belongs to S, solve y in Z_{MS} using the Chinese Remainder Theorem.

2) Compute the secret as: $d = y$ mod m_0.

According to Chinese Remainder Theorem, y can be determined uniquely in Z_{MS}. Since $y < M \leq M_S$, the solution is also unique in Z_M.

2.4 Thien and Lin's Secret Sharing Scheme

Thien and Lin proposed a (k, n)-threshold-based image secret sharing scheme [18] by cleverly using Shamir's SSS [16] to generate shares. The essential idea is to use a polynomial function of order $(k-1)$ to construct n image shares from a $L \times L$ pixel secret image (denoted as I) as

$$\begin{aligned} S_x &= I(i_{k+1}, j) + I(i_{k+2}, j)x + I(i_{k+3}, j)x^2 \\ &\quad +... + I(i_{k+k}, j)x^{k-1} \text{ mod } p, \end{aligned} \tag{1}$$

where $0 \leq i \leq L/k$ and $1 \leq j \leq L$. This method reduces the size of image shares to become 1/k of the size of the size of the secret image. Any k image shares are able to reconstruct every pixel value in the secret image.

In above secret sharing schemes, each share contains the complete secret information in encrypted or ciphered form. Apart from above schemes, the idea behind our proposed scheme is that every share has some bytes missing and these missing bytes can be recovered from a set of exactly k shares. Thus a given byte position can be confirmed for any k shares, but not less than k. Hence k shares are required to give back the secret. Here we use image file as a secret data, but it is equally applicable for any digital data. In our first work [12], we proposed the basic concept of our scheme. Where we have shown, a secret can be shared among a set of participants by information sharing that means all the shares contain partial information about the secret. Then we have applied this scheme [14] for audio file by applying intermediate encryption using the digest of a given key and share the header information by applying simple ANDing with individual mask. But in example (Section 3.3), we have shown that if we apply simple ANDing with individual mask, some header information will be opened for attacker. Then [13] we have used this scheme for a digital image by sharing header with the concept linear geometry, where coefficient values are selected from generated shares. Therefore, if and only if k numbers of legitimate shares come together, then header reconstruction is possible, as a result loss-less secret data will be reconstructed. Here we use the previous scheme in modified form where secret byte selection is randomly depending upon the key from the secret field which provide additional protection of the secret data and we discuss the strength of our scheme by analyzing the scheme in Section 5 with comparing existing schemes. Some additional experimental results are shown in Section 7 and the strength of our scheme is tested using statistical analysis (e.g. histogram analysis and correlation value, etc.) in Section 8, which shows that our scheme is completely prefect and secure secret sharing scheme. To establish the strength of our scheme, we have shown mask generation algorithm and our proposed secret sharing scheme with suitable example.

3 Mask Generation Algorithm

The proposed work is based upon masking which employs ANDing for share generation and ORing the predefined minimal number of shares to reconstruction the original.

3.1 Concept

For better understanding let us consider any secret as a binary bit file (i.e. bit is the smallest unit to work upon, in actual implementation one can consider a byte or group of bytes or group of pixels as the working unit). The secret could be an image, an audio or text etc. We shall decompose the bit file of any size onto n shares in such a way that the original bit file can be reconstructed only ORing any k number of shares where $k \leq$n$\geq$ 2 but in practice we should consider 2 $\leq$k<n$\geq$ 3.

Our basic idea is based on the fact that every share should have some bits missing and those missing bits will be replenished by exactly $(k-1)$ other shares but not less than that. So every individual bit will be missed from exactly $(k-1)$ shares and must be present in all remaining (n-k+1) shares, thus the bit under consideration is available in any set of k shares but not guaranteed in less than k shares. Now for a group of bits, for a particular bit position, $(k-1)$ number of shares should have the bit missed and $(n-k+1)$ number of shares should have the bit present and similarly for different positions there should be different combinations of $(k-1)$ shares having the bits missed and $(n-k+1)$ number of shares having the bits present. Clearly for every bit position there should be C^n_{k-1} such combinations and in our scheme thus forms the mask of size C^n_{k-1}, which will be repeatedly ANDed over the secret in any regular order. Different masks will produce different shares from the secret. Thus 0 on the mask will eliminate the bit from the secret and 1 in the mask will retain the bit forming one share. Different masks having different 1 and 0 distributions will thus generate different shares.

Next just ORing any k number of shares we get the secret back but individual share having random numbers of $1'$s and $0'$s reflect no idea about the secret.

A possible set of masks for 5 shares with threshold of 3 shares is shown below:

Share-1:	1	0	1	0	1	0	1	0	1	1
Share-2:	1	0	1	1	1	1	0	1	0	0
Share-3:	1	1	0	0	0	1	1	1	1	0
Share-4:	0	1	1	1	0	0	1	1	0	1
Share-5:	0	1	0	1	1	1	0	0	1	1

One can easily check that ORing any three or more shares we get all $1'$s but with less than three shares some positions still have $0'$s, i.e. remain missing.

3.2 Algorithm

Here we are presenting the algorithm for designing the masks for n shares with threshold k.

Step 1. List all row vectors of size n having the combination of $(k-1)$ numbers of $0'$s and $(n-k+1)$ numbers of $1'$s and arranged them in some predefined order in terms of their decimal equivalent and finally organized them in the form of a matrix. Obvious dimension of the matrix will be $C^n_{k-1} \times n$.

Step 2. Transpose the matrix generated in Step-1. Obvious dimension of the transposed matrix will be $n \times C^n_{k-1}$. Each row of this matrix will be the individual mask for n different shares. The size of each mask is C^n_{k-1} bits, i.e. the size of the mask varies with the value of n and k.

Let us consider the previous example where $n = 5$ and $k = 3$.

Step 1. List of row vectors of size 5 bits with 2 numbers of $0'$s and 3 numbers of $1'$s, arranged in predefined manner as agreed during sharing phase in order to get masks identical to those used in share generation phase. (Here the arrangement is the highest followed by lowest then next highest followed by next lowest and so on, which appears here as 28, 7, 26, 11, 25, 13, 22, 14, 21, 19).

$$\begin{bmatrix} 1&1&1&0&0\\ 0&0&1&1&1\\ 1&1&0&1&0\\ 0&1&0&1&1\\ 1&1&0&0&1\\ 0&1&1&0&1\\ 1&0&1&1&0\\ 0&1&1&1&0\\ 1&0&1&0&1\\ 1&0&0&1&1 \end{bmatrix} C^5_2 \times 5 = 10 \times 5$$

Step 2. Take the transpose of the above matrix and we get the desired masks for five shares as listed above in the form of matrix of dimension $5 \times C^5_2$, i.e. 5×10. There are five masks each of size 10 bits.

Algorithm 1 Pseudo Code for mask generation

```
1: Input: $n$, $k$
2: Output: $mask[n][]$
3: Integer $mask\_generator(n, k, mask[n][])$
4: {
5: $bin\_arr[][n]$: Integer array;
6: $mask\_pattern\_len = 0$;
7: $max\_val = 2^n - 1$;
   // calculate decimal value of $n$ numbers of 1s.
8: for $i = max\_val$ to 1 do
9:     $Decimal\_to\_Binary(i, bin[][])$;
       // calculate binary equivalent of decimal $i$ and store in $bin\_arr[][]$ array.
10:    if ($Zero\_Check(bin[mask\_pattern\_len[n], k))$) then
11:        $mask\_pattern\_len = mask\_pattern\_len + 1$;
           // check whether $(k-1)$ nos. of zero exist or not, if exist then increment $mask\_pattern\_len$ by 1.
12:    end if
13: end for
14: $Rearrange\_Array(bin)$;
    // rearrange the row of $bin[][n]$ array.
15: $Transpose(mask, bin)$;
    //take transpose matrix of $bin[][n]$ and store in $mask[n][]$.
16: Return $mask\_pattern\_len$;
17: }
18: End
```

3.3 Example

Consider a secret message (M) is WBPRACSE27 and the size of M is 10 bytes. Now by applying logical ANDing with individual mask following shares are generated.

S_i	$Mask$	$Shared\ Message$
$Share-1:$	1010101011	$W0P0A0S027$
$Share-2:$	1011110100	$W0PRAC0E00$
$Share-3:$	1100011110	$WB000CSE20$
$Share-4:$	0111001101	$0BPR00CS07$
$Share-5:$	0101110011	$0B0RAC0027$

From above shares we can easily notice that each share contains partial secret information. That is a secret byte corresponding to one in the mask is kept as it is and the secret byte corresponding to zero in the mask is kept as zero. So every share has some bytes missing and these missing bytes can be recovered from a set of exactly k shares. In mask generation algorithm for n shares and k threshold, size of each mask is C^n_{k-1}, where the number of zeros and ones are C^{n-1}_{k-2} and C^{n-1}_{n-k} respectively. Total size of all shares is 50 bytes ($\sum_{i=1}^{n} Sizeof(S_i)$).

Therefore, each share contains C^{n-1}_{n-k} numbers of partial secret bytes for a set of C^n_{k-1} numbers of secret bytes. Now all zero bytes corresponding to zero bit in the mask are discarded, that introduced a unique compression technique [Section 6]. Therefore, above shared message becomes compressed message.

S_i	$Compressed\ Message$
$Share-1:$	$WPAS27$
$Share-2:$	$WPRACE$
$Share-3:$	$WBCSE2$
$Share-4:$	$BPRCS7$
$Share-5:$	$BRAC27$

Now the total size of all shares is 30 bytes (< 50). Here all secret data are partially open to the participants. To overcome this problem encrypts individual share using a key. After that, the key itself is shared and concatenated with individual share to generate complete shares. Details of this scheme are discussed in the following section.

4 Secret Image Sharing Protocol

Our proposed scheme shares both secret data and header structure including key. Therefore every share has two parts secret share and header share.

4.1 Concept

Our proposed scheme is key based secure threshold cryptography. Initially a 16-byte digest string is generated from user given variable length key (UK_y) using MD5. This 16 bytes digest string is used as encryption key (K_y). The concept is variable length key becomes fixed length key. The length of UK_y should be greater than or equal to 16 bytes. Consider UK_y is "testkey@encry185" then generated fixed length K_y is shown in hex form as "CC7B269F18A6DDB8255EAF4799982131" and the length of K_y is 16 bytes. Here we use MD5 but one can use any strong hash function or random number generator. An advantage of using key based encryption is that it provides authentication as long as the key stays secret. It allows encryption and decryption using same key that is symmetric encryption. This scheme is free to save the key, because the key is also shared among the set of participants. So it reduces the chance of compromising. Also, depending upon k and n, n number of masks are generated and each mask is used to generate individual share and a secret byte (SB) corresponding to one in the mask is kept by encrypting using Equation (3). Also the secret byte (SB) corresponding to zero in the mask is simply discarded. Therefore, every share is compressed. Each share has some bytes missing, all missing bytes can be recovered from a set of exactly k shares. Here each SB is selected randomly from secret field using the following equation

$$f(x) = (Z^2 + c) \bmod L. \tag{2}$$

Where L is total number of secret bytes, Z is random value from K_y and c is $f(x-1)$. Therefore, it provides additional protection of the secret data. Here a complete header structure is to be generated using n, k, K_y and the total number of secret bytes in secret. After that header information is shared using Equation (4) and then each shared header (with individual share number) is appended with secret share to generate a complete share. In reconstruction phase, first collect k number of shares and then reconstruct the complete header information. Now from reconstructed header, the value of n and k are used to generate same mask as sharing phase using same mask generation algorithm. Then apply Equation (5) to decrypt shared bytes, which are selected from shares. This is applicable for nonzero (one) value of that mask with same index position. Thus the missing byte is recovered by inserting zero corresponding to zero in the same index position of that mask. Now apply ORing of k numbers of reconstructed bytes to generate the original secret byte. Therefore the missing bytes can be recovered from a set of exactly k shares. After that, each reconstructed secret bytes are placed in proper position to reconstruct the secret as lossless manner. Now stepwise sharing and reconstruction phases are discussed in following section. The conceptual view of our scheme is described in Figure 2.

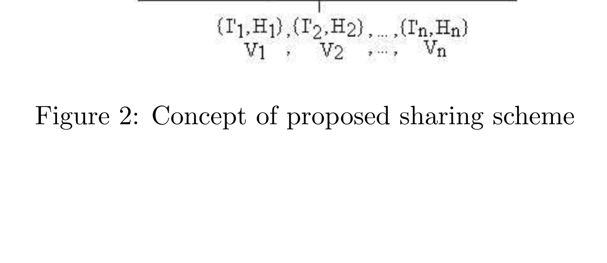

Information Sharing (Using Masking)
Header Construction { Key, n, k, ... }
I1 , I2 ,..., In
Secret Image [S]
[Key]
Encryptor
Sharing
I'1 , I'2 ,..., I'n
H1, H2, ... Hn
Complete Share Generation
{I'1,H1},{I'2,H2}, ... ,{I'n,Hn}
V1 , V2 ,..., Vn

Figure 2: Concept of proposed sharing scheme

4.2 Sharing Phase

The sharing phase of the proposed scheme is stated in the following steps.

Input: threshold (k), total number of share (n), user given variable length key (UK_y), secret image (SI).

Step 1. Initially generate 16 bytes digest from user given variable length key (UK_y). This 16 bytes digest string is used as encryption key (K_y). Here we use MD5 hash function to generate 16 bytes digest string. So variable length key becomes fixed length key.

Step 2. Now construct Header Structure (h) of five fields and put share number (S_n) in 1^{st} field, total number (n) of shares in 2^{nd} field, threshold number (k) in 3^{rd} field, key (K_y) in 4^{th} field, and the total secret bytes (for image, only consider width) (W) in 5^{th} field (See Table 1).

The size of this header structure is 23-bytes. This structure or size of individual field may vary according to our requirements.

Step 3. Generate n masks for n individual shares using the proposed mask generation algorithm [Section 3.2]. Consider the mask pattern length is ML. Therefore, $ML = mask\_generator(n, k, mask[n][])$.

Step 4. Calculate total number of secret bytes (L) present in SI using height and width (W). For other digital files such as text, audio the total number of secret bytes L will be equal to W (i.e. actual size of secret). Also consider an array of N location say Index[L] and initialize all the location by zero, i.e. Index[L] = {0}, where zero indicates unread secret byte at i^{th} position and one indicates read i^{th} secret byte.

$$if \quad \begin{cases} index[i] = 0, & \text{unread} \\ index[i] = 1, & \text{read} \end{cases}$$

Step 5. Now select specific secret byte (SB) from a random position (PS) using Algorithm 2.

The secret byte (SB) corresponding to zero in the mask is simply discarded. Therefore, each share contains partial secret information and for each retained secret bytes apply Step 6 to generate confused secret bytes.

Step 6. Then the i^{th} retained byte (P_i) is ciphered by the j^{th} byte (K_{y_j}) by the following operation:

$$R_i = P_{i-1} \oplus (P_i \times K_{y_j}) \bmod 251, \tag{3}$$

where $i = 0, 1, 2, \cdots, (L-1)$ and $j = \bmod(i, 16)$ and 251 is largest prime number in 8 bits.

0^{th} retained byte is ciphered by the 0^{th} byte of K_y. $R_0 = P_{-1} \oplus (P_0 \times K_{y_0}) \bmod 251$, where P_{-1} is zero.

Algorithm 2 Secret byte (SB) selection and distribution

```
1:  $PS = 0$;
2:  For $j = 0$ to $(L-1)$
3:  $PS = K_y[j\%16] \times K_y[(j+1)\%16] + PS$;
4:  $PS = PS\%L$;
5:  **while** $(Index[PS]! = 0)$ **do**
6:      $PS = (PS+1)\%L$;
7:  **end while**
8:  Read secret byte (SB) from $PS^{th}$ position;
9:  $Index[PS] = 1$; // 0 for unread and 1 for read
10: For $i = 1$ to $n$
11: **if** $(mask[i][j\%ML] = 1)$ **then**
12:     Apply Step-6 for ciphering the SB;
13:     Write the ciphered byte in $i^{th}$ share;
14: **end if**
15: End For;
16: EndFor;
17: End
```

1^{st} retained byte is ciphered by the 1^{st} byte of K_y. $R_1 = P_0 \oplus (P_1 \times K_{y_1}) \bmod 251$.

2^{nd} retained byte is ciphered by the 2^{nd} byte of K_y. $R_2 = P_1 \oplus (P_2 \times K_{y_2}) \bmod 251$,

...

Same as t^{th} retained byte is ciphered by $(t \bmod 16)_{th}$ byte of K_y. $R_t = P_{t-1} \oplus (P_t \times K_{y_{(t \bmod 16)}}) \bmod 251$.

Step 7. Now from each of n ciphered shares collect k numbers of nonzero sample bytes from prefixed locations and thus matrix (A) of dimension $(n \times k)$ is formed.

$$\begin{bmatrix} a_{[0,0]} & a_{[0,1]} & \cdots & a_{[0,k-2]} & a_{[0,k-1]} \\ a_{[1,0]} & a_{[1,1]} & \cdots & a_{[1,k-2]} & a_{[1,k-1]} \\ . & . & \cdots & . & . \\ . & . & \cdots & . & . \\ . & . & \cdots & . & . \\ a_{[n-2,0]} & a_{[n-2,1]} & \cdots & a_{[n-2,k-2]} & a_{[n-2,k-1]} \\ a_{[n-1,0]} & a_{[n-1,1]} & \cdots & a_{[n-1,k-2]} & a_{[n-1,k-1]} \end{bmatrix}_{\text{n×k}}$$

Step 8. The header (Table 1) excluding the leftmost field is also shared by applying following operation:-

$$V_i = \sum_{j=0,1,\cdots,k-1}^{i=0,1,\cdots,n-1} (a[i,j] \times h[j]). \tag{4}$$

Step 9. Next each header share is appended with the share number (S_n) in the first field and concatenated with the corresponding secret share, which forms one complete share for transmission. (For shared image we have to add extra single height to add the shared header, i.e. if height of SI is h, then shared image height will be $(h+1)$).

Table 1: Header Structure (h)

1-Byte	1-Byte	1-Byte	16-Bytes	4-Bytes
Share number	Total number of Shares	Threshold	Encryption Key	Size of Secret
$[S_n]$	$[n]$	$[k]$	$[K_y]$	$[W]$

4.3 Reconstruction Phase

The reconstruction phase of the proposed scheme is stated in the following steps.

Input: k number of shares.

Step 1. Collect k-numbers of share and extract confused header information. Also generates $(k \times k)$ matrix (A).

Step 2. Now applying any conventional linear equation solving technique to reconstruct the original Header information.

Step 3. Once the original Header is reconstructed, we extract the 16 bytes digest string as well as the encryption key (K_y).

Step 4. Now using n and k, extracted from reconstructed header structure, generate n masks using same mask generation algorithm used in share generation phase (same set of mask in same order is reconstructed at the receiving end, used for expanding the compressed shares).

Step 5. According to the share number of the share holder appropriate mask is used to expand the secret share part by inserting zero bytes corresponding to zero in the corresponding mask.

Step 6. Calculate L using shared image height (h) and extracted W from header structure, i.e. $L = (h - 1) \times W$.

Step 7. Ciphered bytes (R_i) corresponding to 1 position in the mask, have generated by the Equation (3). So, apply following operation to get original byte (P_i).

$$P_i = P_{i-1} \oplus (R_i \times M_j^{-1}) \bmod 251. \tag{5}$$

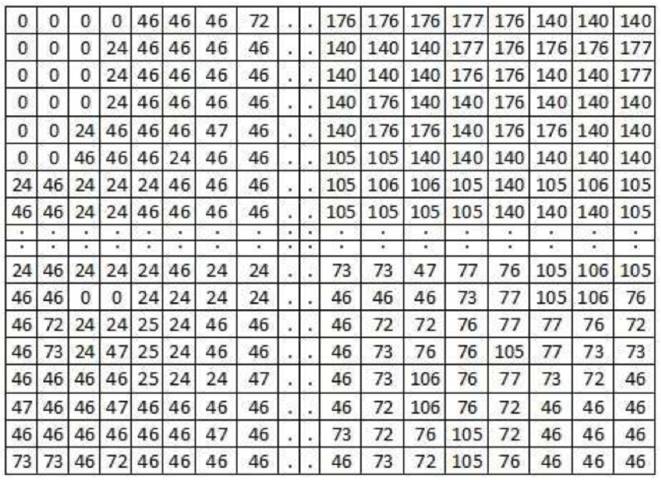
Where M_j^{-1} is the multiplicative inverse of K_{yj}.

Step 8. Now k numbers of P_i are ORed to generate a single secret byte (SB) and placed in PS position using Algorithm 3 (PS generation will be same as to the sharing phase).

Step 9. Secret image is reconstructed in a lossless manner by reconstruction of L numbers of secret bytes.

Algorithm 3 Reconstructed secret byte (SB) writing

```
1: $PS = K_y[i\%16] \times K_y[(i+1)\%16] + PS$;
2: $PS = PS\%L$;
3: while $(Index[PS]! = 0)$ do
4:     $PS = (PS + 1)\%L$;
5: end while
6: Write the secret byte (SB) in $PS^{th}$ position;
7: $Index[PS] = 1$; // 0 for unread and 1 for read
8: End
```



0	0	0	0	46	46	46	72	.	.	176	176	176	177	176	140	140	140
0	0	0	24	46	46	46	46	.	.	140	140	140	177	176	176	176	177
0	0	0	24	46	46	46	46	.	.	140	140	140	176	176	140	140	177
0	0	0	24	46	46	46	46	.	.	140	176	140	140	176	140	140	140
0	0	24	46	46	46	47	46	.	.	140	176	176	140	176	176	140	140
0	0	46	46	46	24	46	46	.	.	105	105	140	140	140	140	140	140
24	46	24	24	24	46	46	46	.	.	105	106	106	105	140	105	106	105
46	46	24	24	46	46	46	46	.	.	105	105	105	105	140	140	140	105
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
24	46	24	24	24	46	24	24	.	.	73	73	47	77	76	105	106	105
46	46	0	0	24	24	24	24	.	.	46	46	46	73	77	105	106	76
46	72	24	24	25	24	46	46	.	.	46	72	72	76	77	77	76	72
46	73	24	47	25	24	46	46	.	.	46	73	76	76	105	77	73	73
46	46	46	46	25	24	24	47	.	.	46	73	106	76	77	73	72	46
47	46	46	47	46	46	46	46	.	.	46	72	106	76	72	46	46	46
46	46	46	46	46	46	47	46	.	.	73	72	76	105	72	46	46	46
73	73	46	72	46	46	46	46	.	.	46	73	72	105	76	46	46	46

Figure 3: Secret Image: Size (61×52)

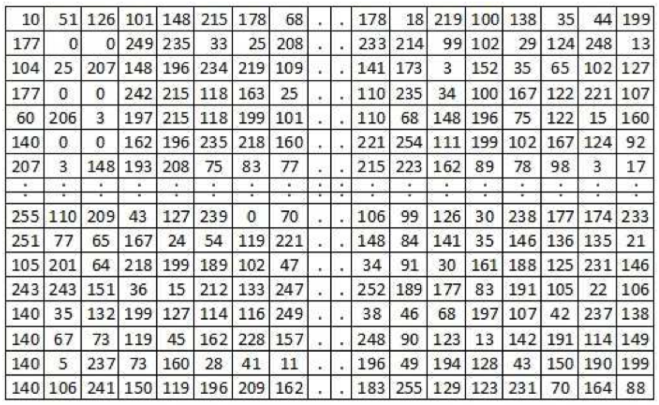
4.4 Example

Here we discuss the sharing phase of our proposed scheme using an example. Consider an image of height and width are 52 and 61 respectively (See Figure 3).

Here we use (3, 5)-threshold sharing scheme with the key $K_y = \{42, 74, 98, 119, 50, 68, 47, 180, 137, 245, 201, 168, 67, 254, 105, 254\}$.

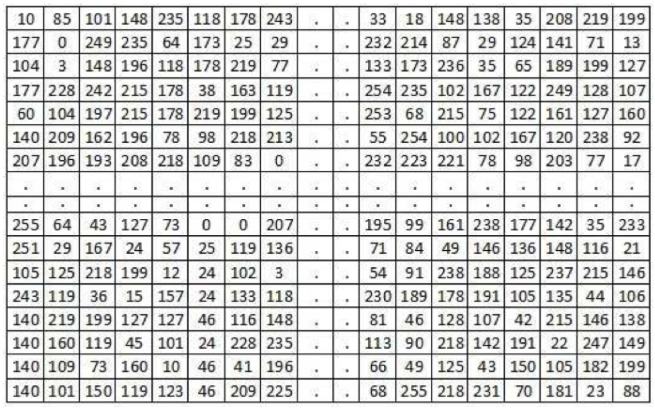
Now select a specific secret byte from a random field among 3172 bytes (61×52), which will be selected depending upon the K_y using Equation (2). Figures 4, 5, and 6 show different share after applying intermediate operation. These figures show each share contains partial confused secret information. That provides additional protection of the secret data. Now generate matrix (A), by taking first k number of nonzero fixed sample values from n shares. Here we consider first non-zero bytes. One can take any non-zero sample bytes from encrypted shares, but it should be same for both sharing and reconstruction phases.

$$A = \begin{array}{c} I_1^{'} \\ I_2^{'} \\ I_3^{'} \\ I_4^{'} \\ I_5^{'} \end{array} \begin{bmatrix} 10 & 51 & 126 \\ . & . & . \\ 10 & 85 & 101 \\ . & . & . \\ 126 & 85 & 148 \end{bmatrix}$$



10	51	126	101	148	215	178	68	.	.	178	18	219	100	138	35	44	199
177	0	0	249	235	33	25	208	.	.	233	214	99	102	29	124	248	13
104	25	207	148	196	234	219	109	.	.	141	173	3	152	35	65	102	127
177	0	0	242	215	118	163	25	.	.	110	235	34	100	167	122	221	107
60	206	3	197	215	118	199	101	.	.	110	68	148	196	75	122	15	160
140	0	0	162	196	235	218	160	.	.	221	254	111	199	102	167	124	92
207	3	148	193	208	75	83	77	.	.	215	223	162	89	78	98	3	17
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
255	110	209	43	127	239	0	70	.	.	106	99	126	30	238	177	174	233
251	77	65	167	24	54	119	221	.	.	148	84	141	35	146	136	135	21
105	201	64	218	199	189	102	47	.	.	34	91	30	161	188	125	231	146
243	243	151	36	15	212	133	247	.	.	252	189	177	83	191	105	22	106
140	35	132	199	127	114	116	249	.	.	38	46	68	197	107	42	237	138
140	67	73	119	45	162	228	157	.	.	248	90	123	13	142	191	114	149
140	5	237	73	160	28	41	11	.	.	196	49	194	128	43	150	190	199
140	106	241	150	119	196	209	162	.	.	183	255	129	123	231	70	164	88

Figure 4: $I_1^{'}$: Size (37×52)



10	85	101	148	235	118	178	243	.	.	33	18	148	138	35	208	219	199
177	0	249	235	64	173	25	29	.	.	232	214	87	29	124	141	71	13
104	3	148	196	118	178	219	77	.	.	133	173	236	35	65	189	199	127
177	228	242	215	178	38	163	119	.	.	254	235	102	167	122	249	128	107
60	104	197	215	178	219	199	125	.	.	253	68	215	75	122	161	127	160
140	209	162	196	78	98	218	213	.	.	55	254	100	102	167	120	238	92
207	196	193	208	218	109	83	0	.	.	232	223	221	78	98	203	77	17
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
255	64	43	127	73	0	0	207	.	.	195	99	161	238	177	142	35	233
251	29	167	24	57	25	119	136	.	.	71	84	49	146	136	148	116	21
105	125	218	199	12	24	102	3	.	.	54	91	238	188	125	237	215	146
243	119	36	15	157	24	133	118	.	.	230	189	178	191	105	135	44	106
140	219	199	127	127	46	116	148	.	.	81	46	128	107	42	215	146	138
140	160	119	45	101	24	228	235	.	.	113	90	218	142	191	22	247	149
140	109	73	160	10	46	41	196	.	.	66	49	125	43	150	105	182	199
140	101	150	119	123	46	209	225	.	.	68	255	218	231	70	181	23	88

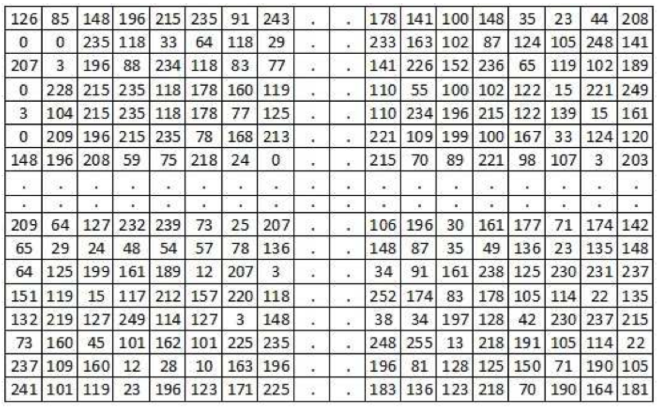
Figure 5: $I_3^{'}$: Size (37×52)

Now generate a header structure as Table 1, here key part contains our encryption key.

$$A = \begin{bmatrix} 10 & 51 & 126 \\ ... & ... & ... \\ 10 & 85 & 101 \\ ... & ... & ... \\ 126 & 85 & 148 \end{bmatrix} \times \begin{bmatrix} 5 \\ 3 \\ 42 \end{bmatrix}$$

$$= \begin{bmatrix} (5495) \\ ... \\ (4547) \\ ... \\ (7101) \end{bmatrix} \Rightarrow \begin{bmatrix} (0 \quad 21 \quad 119) \\ ... \\ (0 \quad 17 \quad 195) \\ ... \\ (0 \quad 27 \quad 189) \end{bmatrix}$$

Therefore shared header information is as Table 2.



126	85	148	196	215	235	91	243	.	.	178	141	100	148	35	23	44	208
0	0	235	118	33	64	118	29	.	.	233	163	102	87	124	105	248	141
207	3	196	88	234	118	83	77	.	.	141	226	152	236	65	119	102	189
0	228	215	235	118	178	160	119	.	.	110	55	100	102	122	15	221	249
3	104	215	235	118	178	77	125	.	.	110	234	196	215	122	139	15	161
0	209	196	215	235	78	168	213	.	.	221	109	199	100	167	33	124	120
148	196	208	59	75	218	24	0	.	.	215	70	89	221	98	107	3	203
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
209	64	127	232	239	73	25	207	.	.	106	196	30	161	177	71	174	142
65	29	24	48	54	57	78	136	.	.	148	87	35	49	136	23	135	148
64	125	199	161	189	12	207	3	.	.	34	91	161	238	125	230	231	237
151	119	15	117	212	157	220	118	.	.	252	174	83	178	105	114	22	135
132	219	127	249	114	127	3	148	.	.	38	34	197	128	42	230	237	215
73	160	45	101	162	101	225	235	.	.	248	255	13	218	191	105	114	22
237	109	160	12	28	10	163	196	.	.	196	81	128	125	150	71	190	105
241	101	119	23	196	123	171	225	.	.	183	136	123	218	70	190	164	181

Figure 6: $I_5^{'}$: Size (37×52)

Table 2: The shared header information

	$[S_n]$	[Shared Header]
H_1	1	(0 21 119) - - -
	- - -	- - -
H_3	3	(0 17 195) - - -
	- - -	- - -
H_5	5	(0 27 189) - - -

Now, actual shares after concatenation of I_i', V_i are as follows.

$$\begin{aligned} V_1 &= I_1', H_1. \quad \text{Size}(37 \times 53) \\ V_2 &= I_2', H_2. \quad \text{Size}(37 \times 53) \\ &\vdots \quad \vdots \\ V_5 &= I_5', H_5. \quad \text{Size}(37 \times 53). \end{aligned}$$

V_1, V_2, $\cdots$, V_5 are the complete shares for transmission. It shows, if and only if k number of shares are came together, then reconstruction is possible, otherwise reconstructed data will be completely different from original.

5 Analysis of the Protocol

In our algorithm for n shares with threshold k size of each mask is C^n_{k-1} where we have C^{n-1}_{k-2} zero and C^{n-1}_{n-k} ones. Then each share contains C^{n-1}_{n-k} number of bytes for C^n_{k-1} number of bytes of secret image. So percentage of information contain in each share is $(C^{n-1}_{n-k}/C^n_{k-1}) \times 100$. This clearly indicates that higher the number of shares and higher the threshold value, i.e. nearer to the number of shares lesser the content of information in each share.

Table 3 shows percentage of information in each share of proposed scheme and other schemes. Next during logical ANDing of the mask with secret image, the bytes of the image corresponding to the one bits in the mask are retained and the zero bytes corresponding to zero bit in the mask will be collapsed, which finally produces a set

Table 3: Percentage of information in each share for different k, n

n	k	C^{n-1}_{n-k}	C^{n}_{k-1}	Our Scheme	[16]	[2]	[1]
8	8	1	8	12.5	100	100	100
8	7	7	28	25	100	100	100
8	6	21	56	37.5	100	100	100
8	5	35	70	50	100	100	100
6	5	5	15	33.3	100	100	100
6	4	10	20	50	100	100	100
5	3	6	10	60	100	100	100

of scrambled bytes; effectively the retained information in compressed and thus being further ciphered.

Next the scrambled bytes are further ciphered using modulo multiplication techniques with the MD5 digest of the key given at the time of transmission (key). It may be noted that instead of using the key directly we have used the digest of the key for encryption, then the size of the key is immaterial which provides additional strength against cryptanalysis for the key.

Finally here the key may be the session key, i.e. the key will be varied with every transmission and our algorithm is free from key distribution hazard as the key itself is further shared in the secret shares.

Thus from individual shares there is hardly any leakage of information vis-a-vis unless minimal number of untampered share which is not tampered is collected nothing is revealed. Only when minimal number of valid share is collected one can form the key and get the information about total number of shares created. After knowing number of shares and the threshold (which is known) one can form the mask and from the key one can get the digest. From the digest using multiplicative inverse we get the compressed shares and the actual shares using the masks. Finally ORing the shares we get the original secret.

6 Analysis of Compression

All masking pattern has equal number of zeros with different distribution only. In every share we collapse all zero bytes corresponding to zero bit in the corresponding mask. It may be noted that as k is closer to n, more is compression, i.e. maximum for $k = n$.

Next for lossless expanding, knowing n and k we can redesign all n masks using our original mask generation algorithm. According to the share number of the share holder appropriate mask is used to expand the secret share by inserting zero bytes corresponding to zero bit in the corresponding mask.

In our example of (3, 5) the mask size is of 10 bits and every mask has 4 zeroes, thus every secret can be compressed by approximately 40 percent, obviously the compression varies with (k, n). In case of an example of (5, 6) the mask size is 15 bits and every mask has 10 zeroes, thus compression will be 66.6 % (See Table 4 and Figures 7 and 8).

Table 4: Compression rate for different k and n

Threshold (k)	Length of Masking Pattern	Number of zero in masking pattern	Approximate Compression Rate (percent)
2	5	1	20
3	10	4	40
4	10	6	60
5	5	4	80

Total number of Shares (n) = 5.

Threshold (k)	Length of Masking Pattern	Number of zero in masking pattern	Approximate Compression Rate (percent)
2	6	1	16
3	15	5	33
4	20	10	50
5	15	10	66
6	6	5	83

Total number of Shares (n) = 6.

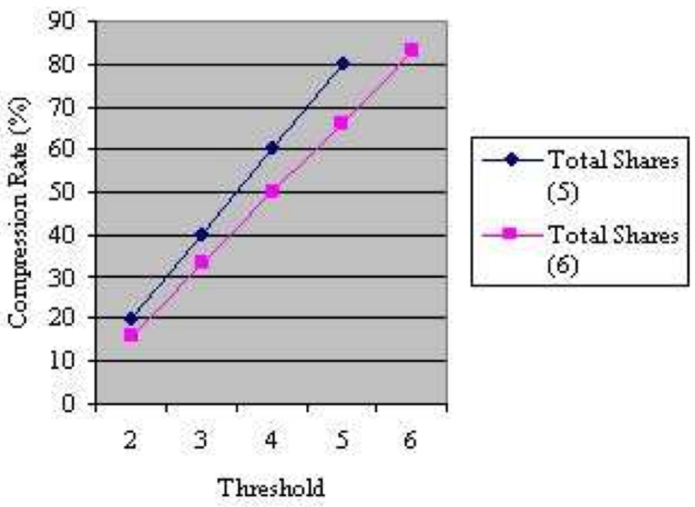
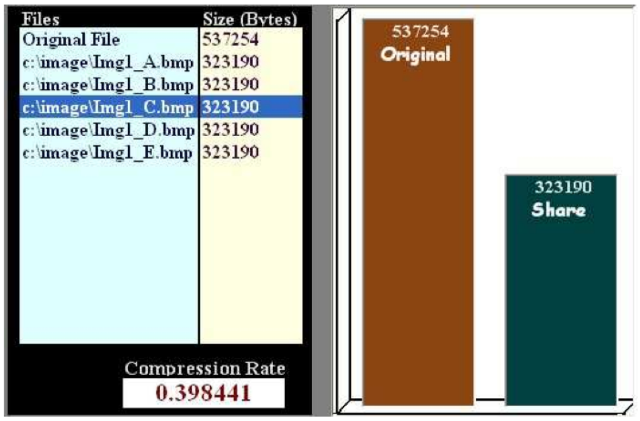


Figure 7: Threshold vs. compression ratio

7 Experimental Result

7.1 Experimental Result for 24-bit bmp Image

Figure 9 shows a secret image (Figure 9(a)) is shared among 5 participants by the user given key (UK_y) "2936451090872310". Here threshold value is 3. At the reconstruction phase, if we collect only 3 or more shares



Files	Size (Bytes)
Original File	537254
c:\image\Img1_A.bmp	323190
c:\image\Img1_B.bmp	323190
c:\image\Img1_C.bmp	323190
c:\image\Img1_D.bmp	323190
c:\image\Img1_E.bmp	323190

Compression Rate
0.398441

Figure 8: Compression rate

then reconstructed secret is lossless, but less than 3 shares are not sufficient to reconstruct the secret data. In the time of reconstruction no need to remember the key, because key as well as secret data is shared among set of participants.

7.2 Experimental Result for Gray Image

Figure 10 shows a secret image (Figure 10(a)) of size 181200 bytes is shared among 5 participants by the user given key "2936451290874310" and the value of k is 3. Here generated shares size are less than secret image and which hold the partial secret information. That provides an additional protection of secret image.

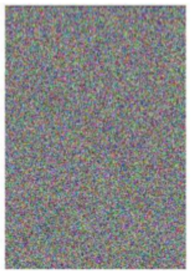
8 Strength and Security Analysis

A secure shared cryptography algorithm should be robust against all types of attacks such as cryptanalytic, statistical. Here we discuss the security analysis of the proposed algorithm by addressing key sensitivity analysis and different statistical analysis. The resistance against different types of attack is useful measure of the performance of a cryptosystem. Therefore some security analysis results are incorporated in the following section to prove the validity of our proposed scheme.

8.1 Key Sensitivity Analysis

In our scheme, shared data is highly sensitive to the secret key. Here user given variable length key (UK_y) is converted as a fixed length key (16 bytes) using MD5 hash function. Here we use MD5, but one can use any hash function or random number generator. Now this fixed length key is used as encryption key (K_y). Generated shares are varied for a single bit/byte changes in the key, because secret bytes are selected randomly from secret field depending upon the key and secret bytes are also encrypted using the key. The merits of our scheme is that key based shared cryptography that introduce the

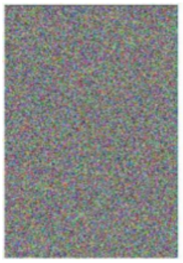
(a). Secret Img1.bmp (453 × 395) Size = 537254 Bytes



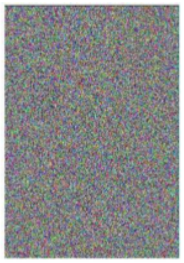
(b). Img1_A.bmp (272 × 396) Size =323190 Bytes

(c). Img1_B.bmp (272 × 396) Size = 323190 Bytes

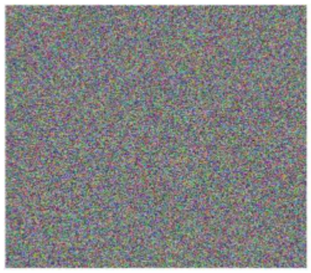
(d). Img1_C.bmp (272 × 396) Size = 323190 Bytes



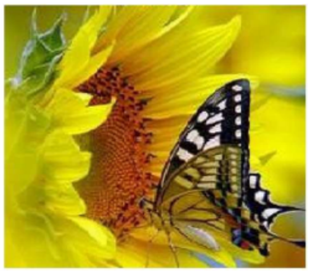
(e). Img1_D.bmp (272 × 396) Size = 323190 Bytes



(f). Img1_E.bmp (272 × 396) Size = 323190 Bytes



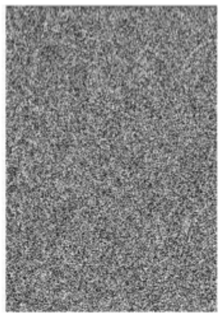
(g). Decode.bmp (Noisy Image Construction using Img1_B.bmp and Img1_E.bmp)

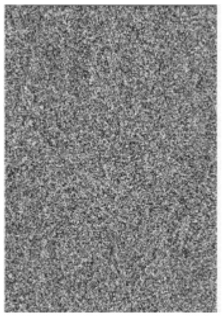


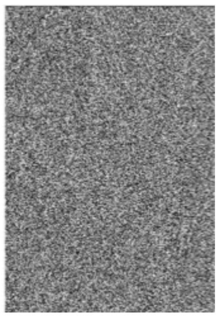
(h). Decode.bmp (Original Image Construction using any three shares)

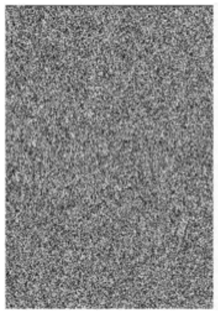
Figure 9: (3, 5)-Sharing and Reconstruction for 24-bit image

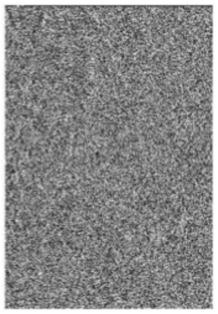
(a). Secret Img2.bmp (453 × 395) Size = 181200 Bytes

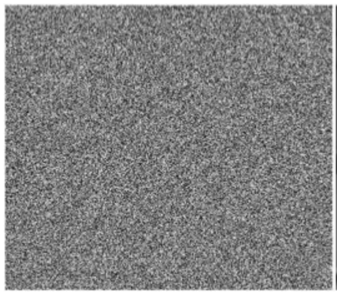

(b). Img2_A.bmp (273 × 396) Size = 110374 Bytes


(c). Img2_B.bmp (273 × 396) Size = 110374 Bytes


(d). Img2_C.bmp (271 × 396) Size = 108790 Bytes


(e). Img2_D.bmp (271 × 396) Size = 108790 Bytes


(f). Img2_E.bmp (271 × 396) Size = 108790 Bytes


(g). Decode.bmp (Noisy Image Construction using Img2_A.bmp and Img2_E.bmp)

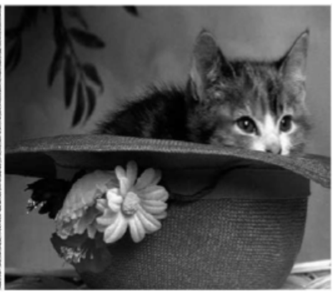
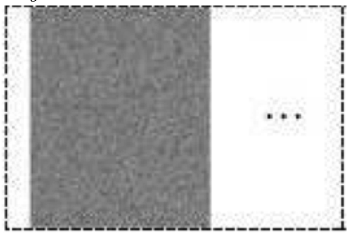

(h). Decode.bmp (Original Image Construction using any three shares)

Figure 10: (3, 5)-Sharing and Reconstruction for gray scale image

concept of avalanche effect and no need to remember the key that overcomes the concept of single point failure, which provides additional protection to the secret data.

(a). Secret Img3.bmp

Group-A with (3, 5) scheme and UK_y: "testkey@encry184"


(b). Share-1

Group-B with (3, 5) scheme and UK_y: "testkey@encry185"

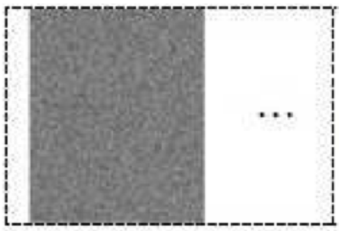

(c). Share-1

Figure 11: Two groups with same secret and different keys

Figure 11 shows a secret image is shared between two groups with different keys using (3, 5) scheme. Among five shares, only first share is shown for each group. Last character indicates the difference between two keys. Here two keys have single byte difference. Correlation value between Figure 11(b) (first share from group-A) and Figure 11(c) (first share from group-B) is 0.0018. This value (closer to zero) indicates two shared images are completely different and there is no such statistical relation. This part tuned an additional protection of secret data. Only when qualified a set of legitimate shares comes together, then reconstruction is possible. Figure 12 shows that a secret data is shared ((3, 5) scheme) among two groups by two different keys. In reconstruction phase, 3 selected shares are A.1, A.3 and A.5 (shares belong to same group) then S is reconstructed whereas if collected shares are 3 but taking from two groups, i.e. A.2, B.3 and A.5, at this situation reconstruction is not possible.

Figure 13 shows collision free random index positions for selecting secret bytes from secret field of size 20 bytes with two users given keys 'testkey@encry184' and 'testkey@encry185'.

8.2 Statistical Analysis

Statistical analysis is crucial importance for a cryptosystem. An ideal cryptosystem should be resistive against

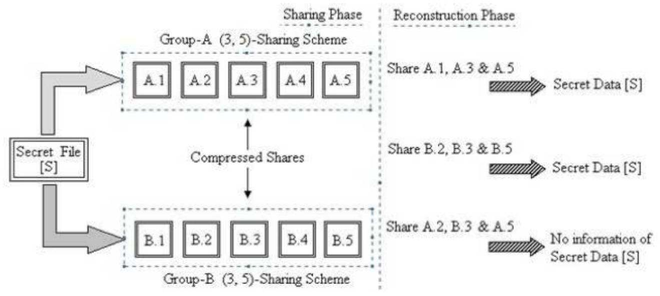


Figure 12: Reconstruction of a Secret file using qualified set of legitimate shares

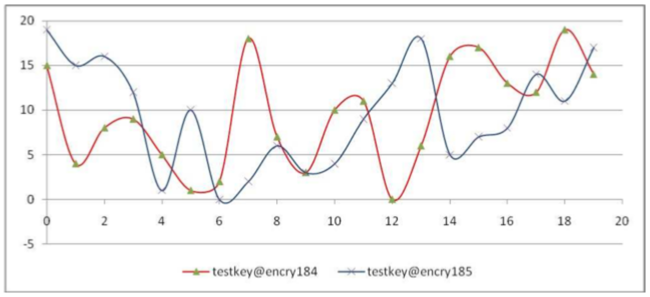


Figure 13: Random index positions for two keys

any statistical attack. To prove the robustness of the proposed algorithm, we have performed the following statistical test such as histogram analysis, correlation analysis, etc.

8.2.1 Histogram Analysis

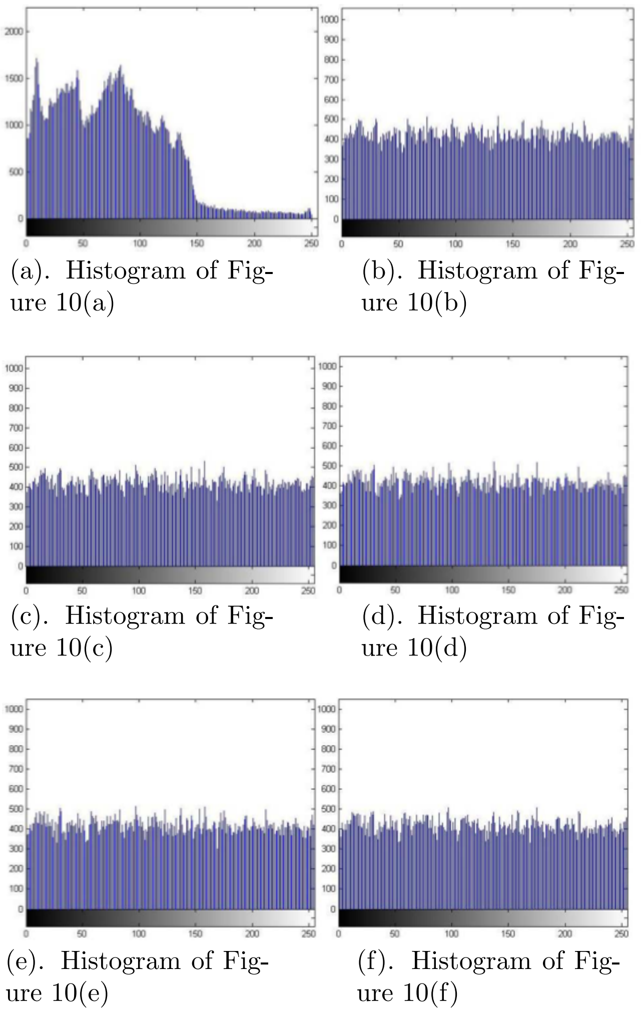
The histogram analysis clarifies how pixels in an image are distributed by plotting the number of pixels at each intensity level. Histogram analysis of gray scale secret image (SecretImg2.bmp) with respect to shared images is shown in Figure 14. The histogram of shared images has uniform distribution which is significantly different from the original image and has no statistical similarity in appearance (See Figure 14).

8.2.2 Correlation Value

A secret shared cryptography scheme must generate shared images independent of the original secret image. Therefore, they must have a very low correlation coefficient value. Here, we have calculated the correlation between the shares. Also the correlation between original and reconstructed image is shown in Table 5. The correlation value is calculated using Equation (6).

$$r = \frac{\sum_m \sum_n (A_{mn} - A')(B_{mn} - B')}{\sqrt{(\sum_m \sum_n (A_{mn} - A')^2 \sum_m \sum_n (B_{mn} - B')^2)}}, \quad (6)$$

where A' and B' are mean of A and B respectively. A low value of correlation coefficient shows that there is no straight relation between the original and encrypted images. Here generated shared images are compressed. So it is impossible to show the correlation value between secret image and shared images. Following table (Table 5) shows the correlation value among shared images and secret image and reconstructed images.



(a). Histogram of Figure 10(a) (b). Histogram of Figure 10(b)

(c). Histogram of Figure 10(c) (d). Histogram of Figure 10(d)

(e). Histogram of Figure 10(e) (f). Histogram of Figure 10(f)

Figure 14: Histogram of Secret image (Figure 10(a)) and Shared images

Table 5: Correlation value

SL No.	Images		Correlation value
1	Figure 10(b) and (c)	:	0.5125
2	Figure 10(d) and (e)	:	0.5057
3	Figure 10(d) and (f)	:	0.1764
4	Figure 10(e) and (f)	:	0.5058
5	Figure 10(a) and (g)	:	-0.0030
6	Figure 10(a) and (h)	:	1

In the above table the 6th entry shows the correlation value between secret image (Figure 10(a)) and reconstructed image (Figure 10(h)) using 3 shares and the value is one, so the reconstruction is lossless. The 5th entry shows a value close to zero for the correlation between secret image (Figure 10(a)) and reconstructed image (Figure 10(g)) using 2 shares.

8.2.3 MSE and PSNR Measure

The Mean Square Error (MSE) and Peak Signal to Noise Ratio (PSNR) for the proposed technique have been computed for different images. The high value of MSE and low value of PSNR cause the resulting encrypted image more randomness. MSE is calculated using the formula

$$MSE = (\Sigma_{i-1}^{N}\Sigma_{j=1}^{M}[C(i,j) - C^{'}(i,j)]^2)/MN,$$

where, $c(I,j)$ and $c^{'}(I,j)$ are the i^{th} row and j^{th} column pixel of two images C and $C^{'}$, respectively. M and N are the number of rows and columns of an image. PSNR can be computed by

$$PSNR = 10 \times log_{10}[R^2/MSE],$$

where R is 255 as grey image has been used in this experiment. Calculated results of MSE and PSNR are tabulated in Table 6.

Table 6: MSE and PSNR value

SL No.	Images		MSE	PSNR
1	Figure 10(b) and (c)	:	115.6822	27.4981
2	Figure 10(d) and (e)	:	117.2003	27.4415
3	Figure 10(d) and (f)	:	194.8199	25.2345
4	Figure 10(e) and (f)	:	117.2881	27.4383
5	Figure 10(a) and (g)	:	233.7130	24.4437
6	Figure 10(a) and (h)	:	0	Infinity

High value MSE and low value PSNR indicate that two images are completely different. On the other hand, the high value of PSNR indicates the high quality image.

9 Conclusions

This paper shows a secured key based secret sharing scheme where key as well as secret data is shared among set of participants. Here image is selected as a secret data, although proposed scheme is strongly applicable for other digital data, such as text, audio, etc. In the sharing phase all secret bytes are selected randomly from secret field depending upon the key and each generated share holds partial secret information in scrambled and encrypted form. That provides additional protection of the secret image and also reduces the bandwidth required for transmission.

In our scheme if and only if numbers of collecting shares are equal to k or more and none of the share is tampered, then only the original secret image is reconstructed; otherwise reconstructed image will be completely ciphered, because fewer shares cannot reconstruct the original header, thus we cannot have either right key (K_y) or the information to construct the correct masking pattern. So our proposed scheme can claim to be a Perfect Secret Sharing (PSS) Scheme. Not only that, if a legitimate group of threshold number of shares comes together (i.e. shares from different groups cannot mix as keys are different), then only the original secret is reconstructed. Moreover, key sensitivity analysis and statistical analysis prove the high acceptability of the proposed algorithm.

Acknowledgments

We are thankful to the department of Computer Science & Engineering of MCKV Institute of Engineering, Regent Education & Research Foundation and Jadavpur University, Kolkata for giving us the platform for planning and developing this work in departmental laboratories.

References

[1] C. Asmuth and J. Bloom, "A modular approach to key safeguarding," *IEEE Transaction on Information Theory*, vol. 29, no. 2, pp. 208–210, 1983.

[2] G. R. Blakley, "Safeguarding cryptographic keys," in *Proceedings of AFIPS International Workshop on Managing Requirements Knowledge*, pp. 313, 1979.

[3] K. Y. Chao and J. C. Lin, "Secret image sharing: a boolean-operations based approach combining benefits of polynomial-based and fast approaches," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 2, pp. 263–285, 2009.

[4] Y. Desmedt, "Some recent research aspects of threshold cryptography," in *Proceedings of 1st International Information Security Workshop (ISW'97)*, pp. 158–173, Ishikawa, Japan, 1997.

[5] Y. Desmedt and Y. Frankel, "Shared generation of authenticators and signatures," in *Advances in Cryptolog (RYPTO'91)*, LNCS 576, pp. 457–469, Springer Verlag, 1992.

[6] Y. Desmedt and Y. Frankel, "Threshold cryptosystems," in *Proceedings on Advances in Cryptology (CRYPTO'89)*, LNCS 435, pp. 307–315, 1990.

[7] Y. Desmedt and Y. Frankel, "Homomorphic zero knowledge threshold schemes over any finite abelian group," *SIAM Journal on Discrete Mathematics*, vol. 7, no. 4, pp. 667–675, 1994.

[8] L. Dong and M. Ku, "Novel (n, n) secret image sharing scheme based on addition," in *Sixth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'10)*, pp. 583–586, 2010.

[9] L. Dong, D. Wang, M. Ku, and Y. Dai, "$(2, n)$ secret image sharing scheme with ideal contrast," in *International Conference on Computational Intelligence and Security (CIS'10)*, pp. 421–424, 2010.

[10] H. F. Huang and C. C. Chang, "A novel efficient (t, n) threshold proxy signature scheme," *Information Sciences*, vol. 176, no. 10, pp. 1338–1349, 2006.

[11] E. D. Karnin, J. W. Greene, and M. E. Hellman, "On secret sharing systems," *IEEE Transactions on Information Theory*, vol. IT-29, no. 1, pp. 35–41, 1983.

[12] P. K. Naskar, A. Chaudhuri, D. Basu, and A. Chaudhuri, "A novel image secret sharing scheme," in *Second International Conference on Emerging Applications of Information Technology (EAIT'11)*, pp. 177–180, 2011.

[13] P. K. Naskar, H. N. Khan, U. Roy, A. Chaudhuri, and A. Chaudhuri, "Secret image sharing with embedded session key," in *Computer Information Systems Analysis and Technologies (CISIM'11)*, Communications in Computer and Information Science, vol. 245, pp 286-294, 2011.

[14] P. K. Naskar, H. N. Khan, U. Roy, A. Chaudhuri, and A. Chaudhuri, "Shared cryptography with embedded session key for secret audio," *International Journal of Computer Applications*, vol. 26, no. 8, pp. 5–9, 2011.

[15] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung, "How to share a function securely?," in *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (STOC'94)*, pp. 522–533, 1994.

[16] A. Shamir, "How to share a secret?," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[17] V. Shoup, "Practical threshold signatures," in *Proceedings of Eurocrypt'00*, LNCS 1807, pp. 207–220, Springer-Verlag, 2000.

[18] C. C. Thien and J. C. Lin, "Secret image sharing," *Computers and Graphics*, vol. 26, no. 5, pp. 765–770, 2002.

Prabir Kr. Naskar, B.Tech from Govt. College of Engineering & Leather Technology (WBUT), West Bengal, India and M.Tech from Jadavpur University, West Bengal, India, is presently working as Assistant Professor in the Department of Computer Science & Engineering, MCKV Institute of Engineering, West Bengal, India. Currently he is doing his research work at Jadavpur University, West Bengal, India. His current research interests include: cryptography, information sharing, steganography, watermarking and image processing.

Hari Narayan Khan is an Assistant Professor in the Department of Computer Science & Engineering, Regent Education & Research Foundation, West Bengal University of Technology, Kolkata, India and presently pursuing his research work at Jadavpur University, Kolkata, West Bengal, India. He completed his M.Tech in Computer Technology at Jadavpur University, Kolkata, India. He completed his B.Tech in Electronics & Communication Engineering from Institute of Technology & Marine Engineering under West Bengal University of Technology, Kolkata, India. His research interest includes cryptography, network security, information sharing, steganography and watermarking.

Prof. Atal Chaudhuri, B.E., M.E. & PhD from Jadavpur University, West Bengal, India, is working in the Department of Computer Science & Engineering, Jadavpur University, West Bengal, India for last 29 years. His current research interests include: embedded system, cryptography, information sharing, steganography, watermarking and data mining.