A Framework of Color Image Sharing and Implementation Based on Number Theory

Chaur-Chin Chen *
Institute of Information Systems & Applications
National Tsing Hua University, Hsinchu 30013, Taiwan
E-mail: cchen@cs.nthu.edu.tw

Abstract−Image Sharing is a technique based on secret message sharing. This technical report discusses 3 image sharing techniques based on (1) Shamir [5], (2) Blakley [2], (3) Chinese Remainder Theorem (CRT) [10] which are all based on modular arithmetics of Number Theory. We provide algorithmic approaches with Matlab/C programs for implementations of (k,n)=(4,6)-threshold examples.

Index Terms−*Blakley, Chinese Remainder Theorem (CRT), Image Sharing, Shamir.*

1. INTRODUCTION

Most of the contemporary image sharing techniques rely on one of the three secret sharing strategies migrated from cryptography, including

(1) Shamir strategy [5, 8] which is based on the concept that a plane polynomial curve of degree k-1 can be uniquely constructed if there are at least k distinct points of the curve are provided. The strategy is processed under modular arithmetic computations.

(2) Blakley strategy [2, 3] is based on the concept that seeking a unique intersecting point, given at least k designated nonparalell planes in a k-dimensional vector space.

(3) The strategy of image sharing based on Chinese remainder theorem (CRT) [1, 6, 7, 10, 11] is to designate a set of simultaneous congruence equations to distribute the information (integers) from each individual pixel into participants such that a secret image can be revealed pixel by pixel by collecting at least k information out of n participants.

*This work is supported by Taiwan MOST Grant 104-2221-E-007-096-MY2

The experiments of the aforementioned approaches only show the limited results of small $2 \leq k < n \leq 5$. The issues of computation and implementation under contemporary computers are ignorant or not even mentioned. This report aims to tackle the above shortage. We shall report experimental results of $(k, n) = (4, 6) - threshold$ techniques by using Matlab programming with the Matlab codes being listed. The parameters will also be explicitly released. In other words, the user-friendly Matlab codes for color image sharing and recovering will be reported.

2. SHARING and RECOVERING ALGORITHMS

2.1. Shamir-Based Method [5, 8]

(A) Image sharing algorithm (over n participants)

(1*) Randomly shuffle the pixels of the secret image and pick up a $k < n$, for example, $k = 4,\ n = 6$ in the implementations through this report.

(2) Let $p = 251$, the largest prime less than 255, and suppress all pixels whose gray values larger than 250 to 250.

(3) Sequentially take k pixels of the image which have not been used whose corresponding intensity values are $g_0, g_1, \cdots, g_{k-1}$ to establish a polynomial of degree $k-1$ under modular p operations as

$$f(x) = g_0 + g_1 x + g_2 x^2 + \cdots + g_{k-1} x^{k-1} \ mod\ p \quad (1)$$

Then,
generate n pixel values $f(x_1), f(x_2), \cdots, f(x_n)$ to distribute for the same location of the n shadow images, where $1 \leq x_1 < x_2 < \cdots < x_n \leq 250$.

(4) Repeat step (3) until all pixels of the image are processed.

Note that step (1) is applied to avoid the information of the secret image, for example, the shape, being exposed. This step is sometimes not necessary and could be ignored.

(B) Image recovering algorithm

(1) Collect any k out of n shadow images.

(2) Take the first unused pixel (usually the top-leftmost one), from each of the k shadow images, say, $y_1,\ y_2,\ \cdots\ y_k$,

(3) Use these $\{y_k' s\}$ pixels and the corresponding assigned values $x_1,\ x_2,\ \cdots\ x_k$ for the k collected shadows to recover the corresponding pixel values $g_0,\ g_1,\ \cdots\ g_{k-1}$ of the original image by solving the following linear system of equations with the coefficient matrix being a $k \times k$ Vandermonde matrix.

$$\begin{bmatrix} 1 & x_1 & \cdots & x_1^{k-1} \\ 1 & x_2 & \cdots & x_2^{k-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_k & \cdots & x_k^{k-1} \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_k \end{bmatrix} \quad (2)$$

(4) Repeat steps (2) and (3) until all pixels of the k shadow images have been processed.

(5*) Inversely permute the pixels of the permuted image to recover the original image if step (1) was applied in the sharing process.

2.2. Blakley Geometry-Based Approach [2]

(A) Image sharing algorithm (over n participants)

(1) Let $p = 251$, sequentially take k pixels of the image which have not been used, whose corresponding intensity values are $g_0, g_1, \cdots, g_{k-1}$, to establish a linear equation (hyperplane) under modular p operations as

$$x_1^{(i)} g_1 + x_2^{(i)} + \cdots + x_k^{(i)} g_k = b^{(i)} \quad mod \quad p \qquad (3)$$

where $\{x_j^{(i)},\ b^{(i)} \mid 1 \leq j \leq k,\ 1 \leq i \leq n\}$ could be randomly assigned as long as the matrix whose elements chosen from the coefficients of $\{x_j^{(\tau(i))},\ 1 \leq j \leq k,\ \tau(i) \in \{1, 2, \cdots, n\}\}$ is nonsingular. Each of the constant terms $b^{(i)},\ 1 \leq i \leq n$ could be treated as a corresponding pixel value of the $i-th$ shadow image.

(2) Repeat step (1) until all pixels of the image are processed.

Note that if we assign the coefficients $\{x_j^{(i)},\ 1 \leq j \leq k,\ 1 \leq i \leq n\}$ as $\{1, x_h, x_h^2, \cdots, x_h^{(k-1)} \mid 1 \leq h \leq n\}$, the Blakley is nothing but a Shamir-Based method. Although Blakley's approach could be viewed as an extention of Shamir-Based method, its computations take much longer time and may not be so practical.

(B) Image recovering algorithm

The reconstruction is nothing but solving a sequence of k linear equations of k variables which are picked from the pixel values of the original image.

2.3. Image Sharing Based on Chinese Remainder Theorem

The Chinese remainder theorem (CRT) [7] is issued to solve a set of simultaneous congruence equations which can be stated as follows. Let $m_1, m_2, \cdots, m_k$ be pairwise coprime positive integers, given nonnegative integers $a_1, a_2, \cdots, a_k$, there exists exactly one solution $x \in [0, m_1 m_2 \cdots m_k)$ for the following simultaneous congruence equations

$$\begin{aligned} x &\equiv a_1\ (mod\ m_1) \\ x &\equiv a_2\ (mod\ m_2) \\ &\vdots \\ x &\equiv a_k\ (mod\ m_k) \end{aligned} \qquad (4)$$

The solution can be obtained by performing the following procedure (CRT).

(1) Denote $M = \prod_{i=1}^{k} m_i$ and let $z_i = M/m_i$ for $1 \leq i \leq k$.

(2) Solve y_i for $y_i z_i \equiv 1\ (mod\ m_i)$ for $1 \leq i \leq k$.

(3) Let $x \equiv (a_1 y_1 z_1 + a_2 y_2 z_2 + \cdots + a_k y_k z_k)\ \ mod\ M$.

(4) Since $x \equiv a_i\ (mod\ m_i)\ \ \forall\ 1 \leq i \leq k$, so is the unique solution.

For example, $x = 39\ (mod\ 2 \cdot 5 \cdot 7)$ is the solution for the following congruence equations.

$$\begin{aligned} x &\equiv 1\ (mod\ 2) \\ x &\equiv 4\ (mod\ 5) \\ x &\equiv 4\ (mod\ 7) \end{aligned} \qquad (5)$$

Based on the Chinese remainder theorem (CRT), Mignotte [11], Asmuth and Bloom [1], Shyu [6], Ulutas et al. [10], Tsai and Chen [9] proposed different implementation methods for solving data and image sharing schemes which are reviewed as follows.

2.3.1. Mignotte Sharing Scheme [11]

Mignotte's sharing scheme uses special Mignotte sequences of positive integers. Let k, n be positive integers such that $2 \leq k \leq n$, A Mignotte sequence is a sequence of positive integers $2 \leq m_1 < m_2 < \cdots < m_n$ such that $gcd(m_i, m_j) = 1, \quad for\ 1 \leq i < j \leq n$ where $m_1 m_2 \cdots m_k > m_{n-k+2} m_{n-k+3} \cdots m_n$.

Mignotte threshold secret sharing scheme can be stated as follows.

(1) Let a secret integer $S \in (\alpha, \beta)$, where $\alpha = m_{n-k+2} m_{n-k+3} \cdots \times m_n$ and $\beta = m_1 m_2 \cdots m_k$.

(2) The share a_i is chosen by computing as $a_i \equiv S\ mod\ m_i$ for $1 \leq i \leq n$.

(3) Collecting at least k distinct shares $a_i's$, the secret S could be revealed by using CRT.

One of the disadvantages for this scheme is that the same secret image pixel values will always be encoded as the same value in a shadow image. Asmuth and Bloom [1] has proposed a method to improve this drawback which is adopted by various researches [6, 10, 9] for the implementation of image sharing and recovering which are reviewed as follows.

2.3.2. Asmuth and Bloom Based Sharing Scheme [1]

Based on the idea of Asmuth and Bloom for data sharing, Shyu and Chen [6] extended the sharing scheme proposed by Mignotte to devise a threshold image sharing scheme which uses a pseudo random number generator (PRNG) with a seed to ensure that the same pixel value is not necessarily encoded as the same number. However, the implementation becomes somewhat tedious and complicated becasue the corresponding number acquired by PRNG of each pixel should be recorded. Ulutas et al. [10] proposed a method to drop the PRNG recording numbers for pixels. The sharing and revealing procedures by Ulutas et al. [10] can be summarized as follows.

(A) A Sharing Procdure

(1) Pick up a set of inteters $\{0 < m_0 < m_1 < \cdots < m_n < 257\}$ subject to

  (a) $gcd(m_i, m_j) = 1 \quad for\ 0 \leq i < j \leq n.$

  (b) $L = m_0 \cdot \prod_{i=1}^{k-1} m_{n+1-i} < M = \prod_{i=1}^{k} m_i.$

(2) Specify an integer $T \in (low, high)$ where $low = \lfloor \frac{L}{m_0} \rfloor$ and $high = \lfloor \frac{M}{m0} \rfloor$, and sequentially take a pixel value p from the secret image and do the following tasks according to a lexicographic order.

  (a) If $p < m_0$, compute

$$y = p + \alpha \cdot m_0 \tag{6}$$

  where α is an integer randomly picked up from $[(T+1), M]$.

  (b) If $p \geq m_0$, compute

$$y = (p - m_0) + \beta \cdot m_0 \tag{7}$$

  where $\beta \in [0, T]$ is randomly picked up.

(3) Compute

$$y_i \equiv y \ \ mod\ m_i \quad for \ \ i = 1, 2, \cdots, n \tag{8}$$

  where y_i is the corresponding pixel value in the $i - th$ shadow image for $1 \leq i \leq n$.

(4) Repeat steps (2) and (3) until all pixels of the secret image are processed. The integer m_i associated with the $i - th$ shadow image is preserved by the $i - th$ participant.

(B) The Recovering Procdure

(1) Collect any k shadow images. Sequentially take the first unused pixel a_i from the $i - th$ shadow image.

(2) Apply the Chinese remainder theorem to solve the following simultaneous equations.

$$\begin{aligned} y &\equiv a_1\ (mod\ m_1) \\ y &\equiv a_2\ (mod\ m_2) \\ &\vdots \\ y &\equiv a_k\ (mod\ m_k) \end{aligned} \tag{9}$$

(3) Compute the random parameter γ, that is, α or β in the sharing procedure by

$$\gamma = \lfloor \frac{y}{m_0} \rfloor \tag{10}$$

The corresponding pixel value of the secret image is $y\ mod\ m_0$ if $\gamma > T$, otherwise, $m_0 + (y\ mod\ m_0)$ if $\gamma \leq T$.

(4) Repeat steps (1∼3) until all of the pixels are revealed.

Note that the random integers α and β are not required in the revealing procedure which improves the work implemented by Shyu and Chen [6]. However, a user-specified threshold integer T must be provided during sharing and revealing procedures.

3. A PROPOSED IMAGE SHARING ALGORITHM BASED ON CRT [4]

We extend the sharing scheme proposed by Ulutas in 2009 based upon Chinese Remainder Theorem (CRT) to designate a (k,n)-threshold secret sharing scheme for digital RGB-color images in a TIFF image file format. In the experiments of this report, we consider each of R,G,B images is shared by using the same sharing method though other sharing strategies could be used. To meet the restrictions of CRT and the pixel range is in [0,255], we store the least significant bits of R,G,B parts in a file without being shared. The most significant 7 bits are right shifted in one position.

(A) A Proposed Sharing Algorithm

(1) Select a set of integers $\{m_0, m_1, m_2, \cdots, m_n\}$ which satisfies $m_0 = 128 < m_1 < m_2 < \cdots < m_n \leq 255\}$ and meets the following two requirements.

(a) $gcd(m_i, m_j) = 1$ for $0 \leq i < j \leq n$.

(b) $L = m_0 \cdot \prod_{i=1}^{k-1} m_{n+1-i} \ < \ M = \prod_{i=1}^{k} m_i$.

(2) Each pixel value x_h of r,g,b signals, respectively is computed according to the following equation

$$y_h = (x_h >> 1) + \alpha \cdot m_0, \quad where\ h = r,\ g,\ or\ b$$

As mentioned before, the least significant bits of pixels are stored separately, and $\alpha \in (0, \lfloor \frac{M}{m_0} \rfloor)$ is a randomly generated integer. The purpose to use α is to avoid that the same value is converted into the identical value in a shadow image, whereas, α is not required during a recovering procedure.

(3) We distribute a shadow pixel value for each participant according to the following modular arithmetic from each y_h value obtained above.

$$\begin{aligned} r_i^{(p)} &\equiv y_r \ mod \ m_i \\ g_i^{(p)} &\equiv y_g \ mod \ m_i \quad for\ i = 1, 2, \cdots, n \qquad (11) \\ b_i^{(p)} &\equiv y_b \ mod \ m_i \end{aligned}$$

Note that a color pixel p in the $i-th$ shadow image is recorded as $[r_i^{(p)},\ g_i^{(p)},\ b_i^{(p)}]$.

(4) Repeat steps (2) and (3) until all of the color pixels $\{(y_r, y_g, y_b)\}$ of the secret image are processed, then m_i is associated with the $i-th$ shadow image is kept by the $i-th$ participant for $i = 1, 2, \cdots, n$.

(B) The Recovering Algorithm

(1) Collect at least k shadow images associated with k $m_j's$, say, $m_1, m_2, \cdots, m_k$ without loss of generality.

(2) Sequentially, take the first pixel, say, w, from each of the k shadow images. We use the k values of $\{r_i^{(w)} | i = 1, 2, \cdots, k\}$ and the Chinese Remainder Theorem (CRT) to resolve $y_r^{(w)}$, the similar action is applied to resolve $y_g^{(w)}$ and $y_b^{(w)}$, respectively.

(3) We reverse the sharing process to reveal the pixel values by

$$x_h = ((y_h \ mod \ m_0) << 1) + t_h^{(w)}, \quad h = r,\ g,\ or\ b$$

where $t_h^{(w)}$ is the least significant bit value (either 0 or 1) pre-stored before processing image sharing.

(4) Repeat steps (2) and (3) until all pixels of k shadow images are processed.

If an approximate secret image could be accepted, the pre-stored least significant bits could be ignored and can be randomly generated during a recovering process.

4. EXPERIMENTS

We demonstrate the experimental results of $(k, n) = (4, 6)$ for CRT-based and Shamir-based methods and list the Matlab codes for each method [14].

4.1. Experimental Results By CRT-Based Method

Chuang et al. [4] have reported experimental results implemented in a Linux-based system on a $(k, n) = (3, 5)$-threshold method [12] with the parameters $m_0 = 128$ and $(m_1, m_2, m_3, m_4, m_5) = (247, 251, 253, 254, 255)$ being selected so that each pixel value in shadow images randomly lie in the range [0,255) although the first shadow image may only contain pixel values in the range [0,247), all of the five shadow images look like noise without leaking any information of the secret image. A color image "Peppers" for sharing and recovering is demonstrated that the shadow images associated with $(m_3, m_4, m_5) = (253, 254, 255)$ were used to exactly reveal the original image in the paper [4].

This report extends $(k, n) = (3, 5)$ to $(k, n) = (4, 6)$ by choosing the parameters $m_0 = 128$ and $m_1 = 241$, $m_2 = 247$, $m_3 = 251$, $m_4 = 253$, $m_5 = 254$, $m_6 = 255$. For $(k, n) = (4, 6$, one immediately encounters a problem of over 32-bit integer multiplication computations. We exploit some Matlab built-in function such as **uint64** to overcome this problem. The color image "Lenna" for sharing and recovering is demonstrated that the shadow images associated with $(m_1, m_2, m_3, m_4) = (241, 247, 251, 253)$ were used to exactly reveal the original image as shown in the following figures. The elapsed time for sharing and recovering in a $(k, n) = (4, 6)$-threshold which is run with a Matlab version 2011b requires approximately 18 seconds. **Matlab Codes can be accessed** via [12].

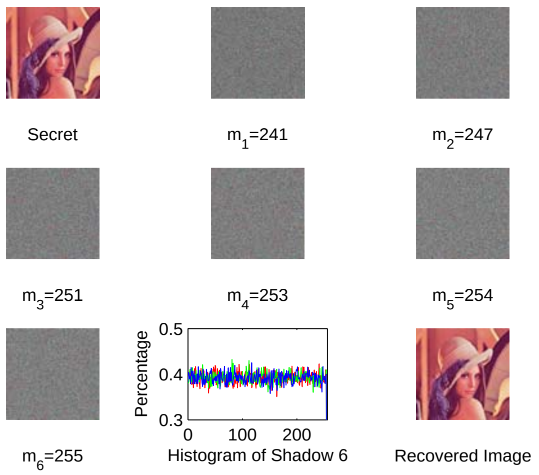

Figure 1: CRT Shadow Images and Reconstruction.

4.2. Experimental Results By Shamir-Based Method

Thien and Lin [8] reported experimental results for $(k, n) = (2, 4)$-threshold bsed on the Shamir polynomial interpolation. We extend the similar work for $(k, n) = (4, 6)$ and tackle the problem by computing the inverse of a 4×4 *Vandermonde matrix* under the modular p=251 arithmetics. The $(m_1, m_2, m_3, m_4, m_5, m_6) = (37, 60, 101, 149, 203, 246)$ are randomly generated from [2,250]. The color image "Mandrill" for sharing and recovering is demonstrated that the shadow images associated with $(m_3, m_4, m_5, m_6) = (101, 149, 203, 246)$ were used to exactly reveal the original image as shown in the following figures.

The elapsed time for sharing and recovering in a $(k, n) = (4, 6)$-threshold which is run with a Matlab version 2011b requires approximately 2 seconds. **Matlab Codes can be accessed via** [13].

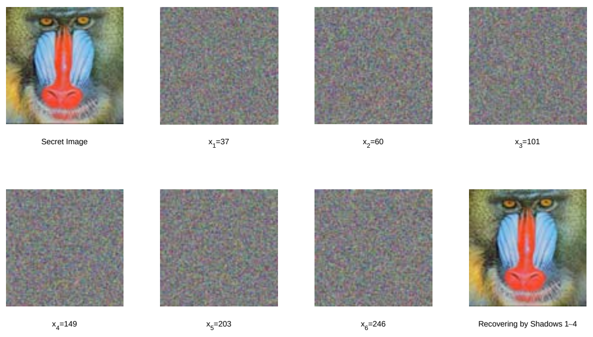

Figure 2: Shamir Shadow Images and Reconstruction.

5. DISCUSSION AND CONCLUSION

This report reviews and discusses commonly used (k, n) threshold methods including Shamir [5, 8], Blakley [2, 3], and CRT-Based methods [1, 10, 11] for image sharing and recovering. Generally speaking, a pixel value falls in [0,255] which should be taken into account in the design for image sharing, based on which, we proposed a simple CRT-based method to implement an image sharing and recovering processes for $(k, n) = (3, 5)$ [4, 12]. Due to the computations for large integers such as those over 32-bit or $2^{31} - 1$ representations, the experiments for $(k, n) = (4, 6)$ have not been reported yet. This report demonstrates the results of sharing and recovering of $(k, n) = (4, 6)$-threshold methods based on Chinese Remainder Theorem and Shamir polynomial interpolation by using simple Matlab codes. The extension for even larger (k, n) or other methods merits further studies.

References

[1] C. Asmuth and J. Bloom, "A Modular Approach to Key Guarding," *IEEE Trans. on Information Theory*, vol. 29, no. 2, 208-210, 1983.

[2] G.R. Blakley, "Safeguarding cryptographic keys," *Proceedings of the National Computer Conference, American Federation of Information Proceeding Societies*, New York, vol. 48, 313-317, 1979.

[3] C. Chen, W.Y. Fu, and C.C. Chen, "A Geometry-Based Image Sharing Approach," *Proceedings of Image and Vision Computing*, Dunedin, Otago, New Zealand, 428-431, 2005.

[4] T.C. Chuang, C.C. Chen, and B. Chien, "Image Sharing and Recovering Based on Chinese Remainder Theorem," *IEEE International Conference on Computer, Consumer, and Control*, 817-820, Xi'an, China, July 4-6, 2016.

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

[6] S.J. Shyu and Y.R. Chen, "Threshold Secret Image Sharing by Chinese Remainder Theorem," *IEEE Asia-Pacific Services Computing Conference*, 1332-1337, Yilan, Taiwan, Dec. 9-12, 2008.

[7] D.R. Stinson, "Cryptography: Theory and Practice," *Champman & Hall / CRC Press*, 2006.

[8] C.C. Thien and J.C. Lin, "Secret image sharing," *Compuers & Graphics*, vol. 26, no. 1, 765-771, 2002.

[9] M.H. Tsai and C.C. Chen, "A Study on Secret Image Sharing," *The Sixth International Workshop on Image Media Quality and Its Applications*, 135-139, Tokyo, Japan, September 12-13, 2013.

[10] M. Ulutas, V.V. Nabiyev, and G. Ulutas, "A New Secret Sharing Technique Based on Asmuth Bloom's Scheme," *IEEE International Conference on Application of Information and Communication Technologies*, 1-5, Baku, Oct. 14-16, 2009.

[11] https://en.wikipedia.org/wiki/ Secret_sharing_using_the_Chinese_remainder_theorem, last access on July 31, 2016.

[12] http://www.cs.nthu.edu.tw/.WWW/CRT2016, last access on July 31, 2017.

[13] http://www.cs.nthu.edu.tw/.WWW/Shamir2017, last access on July 31, 2017.

[14] http://www.cs.nthu.edu.tw/.WWW/NSC2017, last access on July 31, 2017.

Matlab Codes for CRT Sharing and Recovering

```
%% crtNSC.m - CRT-Based (k,n)-threshold sharing and recovering
% Elapsed time = 17.2 seconds in PC/XP running Matlab 2011a version
% (1) Read an RGB-based input image in Tiff format, e.g., Lenna.tiff
%
m=512; n=512;
I=imread('Lenna.tiff');       IC=I;
R=I(:,:,1); G=I(:,:,2); B=I(:,:,3);
% for reconstruction
R0=zeros(m,n,'uint8'); G0=zeros(m,n,'uint8'); B0=zeros(m,n,'uint8');
%
% (2) Image Sharing - generate 6 Shadows for (4,6)-threshold method
%
Shadow=imread('Peppers.tiff'); % Image size used for reconstruction
R1=zeros(m,n,'uint8'); R2=R1; R3=R1; R4=R1; R5=R1; R6=R1;
G1=zeros(m,n,'uint8'); G2=G1; G3=G1; G4=G1; G5=G1; G6=G1;
B1=zeros(m,n,'uint8'); B2=B1; B3=B1; B4=B1; B5=B1; B6=B1;
%
%  (2a) Shift 1 bit to the right for (R,G,B) while
%       preserving the least significant bits
for i=1:m
    for j=1:n
        r0=bitget(R(i,j),1);
        g0=bitget(G(i,j),1);
        b0=bitget(B(i,j),1);
        R(i,j)=(R(i,j)-r0)/2; R0(i,j)=r0;
        G(i,j)=(G(i,j)-g0)/2; G0(i,j)=g0;
        B(i,j)=(B(i,j)-b0)/2; B0(i,j)=b0;
    end
end
%
% (3) Compute the 6 shadow images
%
m0=uint32(128);  S=uint32([241, 247, 251, 253, 254, 255]);
Low=uint32(m0*S(4)*S(5)*S(6)); Up=uint32(S(1)*S(2)*S(3)*S(4));
Lb=round(Low/m0+2);  Ub=round(Up/m0-3);
Rarray=uint32(randi([Lb, Ub],m,n));
y=uint32(1);
for i=1:m
    for j=1:n
        r=R(i,j);  g=G(i,j);  b=B(i,j); t=Rarray(i,j);
        y=t*m0+uint32(r);
        R1(i,j)=mod(y,S(1)); R2(i,j)=mod(y,S(2)); R3(i,j)=mod(y,S(3));
        R4(i,j)=mod(y,S(4)); R5(i,j)=mod(y,S(5)); R6(i,j)=mod(y,S(6));
        y=t*m0+uint32(g);
        G1(i,j)=mod(y,S(1)); G2(i,j)=mod(y,S(2)); G3(i,j)=mod(y,S(3));
        G4(i,j)=mod(y,S(4)); G5(i,j)=mod(y,S(5)); G6(i,j)=mod(y,S(6));
        y=t*m0+uint32(b);
        B1(i,j)=mod(y,S(1)); B2(i,j)=mod(y,S(2)); B3(i,j)=mod(y,S(3));
        B4(i,j)=mod(y,S(4)); B5(i,j)=mod(y,S(5)); B6(i,j)=mod(y,S(6));
    end
end
%
% (4) Print the color input image and its Shadow Color Images
%
S1=Shadow; S2=Shadow; S3=Shadow; S4=Shadow; S5=Shadow; S6=Shadow;
S1(:,:,1)=R1;  S1(:,:,2)=G1;  S1(:,:,3)=B1;  % S1: Color Shadow 1
S2(:,:,1)=R2;  S2(:,:,2)=G2;  S2(:,:,3)=B2;  % S2: Color Sahdow 2
S3(:,:,1)=R3;  S3(:,:,2)=G3;  S3(:,:,3)=B3;  % S3: Color Shadow 3
S4(:,:,1)=R4;  S4(:,:,2)=G4;  S4(:,:,3)=B4;  % S4: Color Shadow 4
S5(:,:,1)=R5;  S5(:,:,2)=G5;  S5(:,:,3)=B5;  % S5: Color Shadow 5
S6(:,:,1)=R6;  S6(:,:,2)=G6;  S6(:,:,3)=B6;  % S5: Color Shadow 6
subplot(3,3,1)
imshow(I)
xlabel('Secret')
subplot(3,3,2)
imshow(S1)
xlabel('m_1=241')
subplot(3,3,3)
imshow(S2)
xlabel('m_2=247')
subplot(3,3,4)
imshow(S3)
xlabel('m_3=251')
subplot(3,3,5)
imshow(S4)
xlabel('m_4=253')
subplot(3,3,6)
imshow(S5)
xlabel('m_5=254')
subplot(3,3,7)
imshow(S6)
xlabel('m_6=255')
imwrite(S1,'shadow1.tif'); imwrite(S2,'shadow2.tif');
imwrite(S3,'shadow3.tif'); imwrite(S4,'shadow4.tif');
imwrite(S5,'shadow5.tif'); imwrite(S6,'shadow6.tif');
```

```
%
% (5) Recovering - from S3~S6 Shadow images to solve the CRT problem
%
% Z=uint64([S(2)*S(3)*S(4),S(1)*S(3)*S(4),S(1)*S(2)*S(4),S(1)*S(2)*S(3)]);
% Y=[80, 12, 204, 65];
%
M=uint64(S(3)*S(4)*S(5)*S(6));
Z=uint64([S(4)*S(5)*S(6),S(3)*S(5)*S(6), S(3)*S(4)*S(6),S(3)*S(4)*S(5)]);
Y=uint64([136, 63, 85, 223]);
A=uint64([255, 255, 255, 255]);
u=uint64(0);   s=uint64(1);
m0=uint64(128);
for i=1:m
    for j=1:n
        A=uint64([(R3(i,j)), R4(i,j), R5(i,j), R6(i,j)]);
        s=sum(A.*(Y.*Z));
        s=mod(s,M);
        R(i,j)=mod(s,m0);
        A=uint64([(G3(i,j)), G4(i,j), G5(i,j), G6(i,j)]);
        s=sum(A.*(Y.*Z));
        s=mod(s,M);
        G(i,j)=mod(s,m0);
        A=uint64([B3(i,j), B4(i,j), B5(i,j), B6(i,j)]);
        s=sum(A.*(Y.*Z));
        s=mod(s,M);
        B(i,j)=mod(s,m0);
    end
end
%
% (6) Print histogram of shadow image 6
%
hr=zeros(256); hg=zeros(256); hb=zeros(256);
for i=1:m
    for j=1:n
        k=R6(i,j)+1; hr(k)=hr(k)+1;
        k=G6(i,j)+1; hg(k)=hg(k)+1;
        k=B6(i,j)+1; hb(k)=hb(k)+1;
    end
end
for k=1:256
    hr(k)=100.0*hr(k)/(m*n);
    hg(k)=100.0*hg(k)/(m*n);
    hb(k)=100.0*hb(k)/(m*n);
end
```

```
subplot(3,3,8)
L=0:255;
plot(L,hr,'r-',L,hg,'g-',L,hb,'b-'); axis([0,256,0.3,0.5])
xlabel('Histogram of Shadow 6')
ylabel('Percentage')
%
% (7) IC holds a perfect revealed image
%
for i=1:m
    for j=1:n
            R0(i,j)=R0(i,j)+2*R(i,j);
            G0(i,j)=G0(i,j)+2*G(i,j);
            B0(i,j)=B0(i,j)+2*B(i,j);
    end
end
IC(:,:,1)=R0; IC(:,:,2)=G0; IC(:,:,3)=B0; imwrite(IC,'tRecov.tif');
subplot(3,3,9)
imshow(IC)
xlabel('Recovered Image')
%title('Reconstruction by Shadows 3\sim6')
```

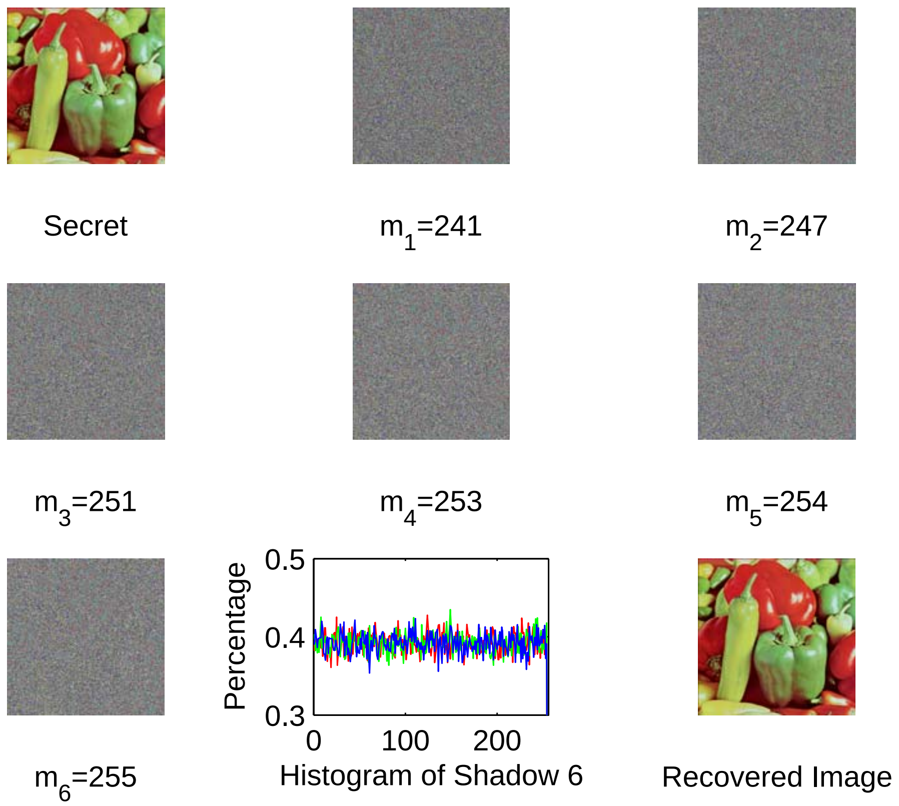

Figure 1: CRT Shadows and Reconstruction for Peppers.

Matlab Codes for Shamir Sharing and Recovering

```
%% shamirNSC.m - Shamir (4,6)-threshold sharing and recovering
%     The Elapsed time ~ 2 seconds for sharing and recovering
% (1) Read an RGB-based input image in Tiff format, e.g., Lenna.tiff
%
m=512; n=512; np=m*n; m2=m/2;  n2=n/2;
I=imread('Mandrill.tiff');        IC=I;
R=I(:,:,1); G=I(:,:,2); B=I(:,:,3);
R0=zeros(m,n); G0=zeros(m,n); B0=zeros(m,n);  % for reconstruction
%
% (2) Establish the table of ax=1 mod p=251; x*f(x)=1 mod p=251
%
p=251;
sinv=zeros(1,256);
for i=1:p-1
    s=i; k=1;
    while (k<p)
        if (mod(s*k,p)==1)
            sinv(i)=k; k=p;
        else
            k=k+1;
        end
    end
end
%
% (3) Image Sharing - generate 6 Shadows for (4,6)-threshold mod p=251
%
Shadow=imread('color256.tif');       % (m2*n2)*3 size for shadows 1~6
R1=zeros(m2,n2); R2=R1; R3=R1; R4=R1; R5=R1; R6=zeros(m2,n2);
G1=zeros(m2,n2); G2=G1; G3=G1; G4=G1; G5=G1; G6=zeros(m2,n2);
B1=zeros(m2,n2); B2=B1; B3=B1; B4=B1; B5=B1; B6=zeros(m2,n2);
%
% (4) Prepare a perfect recovering
%
for i=1:m
    for j=1:n
        if (R(i,j)>250)  R0(i,j)=R(i,j)-250; R(i,j)=250;  end
        if (G(i,j)>250)  G0(i,j)=G(i,j)-250; G(i,j)=250;  end
        if (B(i,j)>250)  B0(i,j)=B(i,j)-250; B(i,j)=250;  end
    end
end
%
```

```
% (4) Compute the shadow images
%
r=zeros(1,4); g=zeros(1,4); b=zeros(1,4);
v=round(250*rand(1,6));
v=sort(v);
v=[37,60,101,149,203,246];   % specified {m_i} for an example only
v2=mod(v.^2,p); v3=mod(v.^3,p);
AX=[1,1,1,1,1,1; v; v2; v3];
for i=0:2:m-2
    ix=1+i/2;
    for j=0:2:n-2
        jy=1+j/2;
        r(1)=R(i+1,j+1); r(2)=R(i+1,j+2); r(3)=R(i+2,j+1); r(4)=R(i+2,j+2);
        yr=mod(r*AX,p);
        R1(ix,jy)=yr(1); R2(ix,jy)=yr(2); R3(ix,jy)=yr(3);
        R4(ix,jy)=yr(4); R5(ix,jy)=yr(5); R6(ix,jy)=yr(6);
        g(1)=G(i+1,j+1); g(2)=G(i+1,j+2); g(3)=G(i+2,j+1); g(4)=G(i+2,j+2);
        yg=mod(g*AX,p);
        G1(ix,jy)=yg(1); G2(ix,jy)=yg(2); G3(ix,jy)=yg(3);
        G4(ix,jy)=yg(4); G5(ix,jy)=yg(5); G6(ix,jy)=yg(6);
        b(1)=B(i+1,j+1); b(2)=B(i+1,j+2); b(3)=B(i+2,j+1); b(4)=B(i+2,j+2);
        yb=mod(b*AX,p);
        B1(ix,jy)=yb(1); B2(ix,jy)=yb(2); B3(ix,jy)=yb(3);
        B4(ix,jy)=yb(4); B5(ix,jy)=yb(5); B6(ix,jy)=yb(6);
    end
end
%
% (5) Print the color input image and its Shadow Color Images
%
S1=Shadow; S2=Shadow; S3=Shadow; S4=Shadow; S5=Shadow; S6=Shadow;
S1(:,:,1)=R1;  S1(:,:,2)=G1;  S1(:,:,3)=B1;  % S1: Color Shadow 1
S2(:,:,1)=R2;  S2(:,:,2)=G2;  S2(:,:,3)=B2;  % S2: Color Sahdow 2
S3(:,:,1)=R3;  S3(:,:,2)=G3;  S3(:,:,3)=B3;  % S3: Color Shadow 3
S4(:,:,1)=R4;  S4(:,:,2)=G4;  S4(:,:,3)=B4;  % S4: Color Shadow 4
S5(:,:,1)=R5;  S5(:,:,2)=G5;  S5(:,:,3)=B5;  % S5: Color Shadow 5
S6(:,:,1)=R6;  S6(:,:,2)=G6;  S6(:,:,3)=B6;  % S5: Color Shadow 6
subplot(2,4,1)
imshow(I)
xlabel('Secret Image')
subplot(2,4,2)
imshow(S1)
xlabel('x_1=37')
subplot(2,4,3)
imshow(S2)
```

```
xlabel('x_2=60')
subplot(2,4,4)
imshow(S3)
xlabel('x_3=101')
subplot(2,4,5)
imshow(S4)
xlabel('x_4=149')
subplot(2,4,6)
imshow(S5)
xlabel('x_5=203')
subplot(2,4,7)
imshow(S6)
xlabel('x_6=246')
%
% (6) Recovering - from S1~S4 Shadow images with v(1~4), respectively
%     The inverse matrix computation may require some preprocessing
%
u=v(1,1:4);
u2=mod(u.^2,p); u3=mod(u.^3,p);
A=[1,1,1,1; u; u2; u3];
ti=mod(round(det(A)),p);
if (ti<0) t=ti+p; else t=ti; end;
s=sinv(t);
C=mod(s*adj(A),p); y=zeros(1,4);
for ix=1:m2
    i=2*(ix-1);
    for jy=1:n2
        j=2*(jy-1);
        y(1)=R1(ix,jy); y(2)=R2(ix,jy); y(3)=R3(ix,jy); y(4)=R4(ix,jy);
        r=mod(y*C,p);
        R(i+1,j+1)=r(1); R(i+1,j+2)=r(2); R(i+2,j+1)=r(3); R(i+2,j+2)=r(4);
        y(1)=G1(ix,jy); y(2)=G2(ix,jy); y(3)=G3(ix,jy); y(4)=G4(ix,jy);
        g=mod(y*C,p);
        G(i+1,j+1)=g(1); G(i+1,j+2)=g(2); G(i+2,j+1)=g(3); G(i+2,j+2)=g(4);
        y(1)=B1(ix,jy); y(2)=B2(ix,jy); y(3)=B3(ix,jy); y(4)=B4(ix,jy);
        b=mod(y*C,p);
        B(i+1,j+1)=b(1); B(i+1,j+2)=b(2); B(i+2,j+1)=b(3); B(i+2,j+2)=b(4);
    end
end
%
% (7) Histogram of Shadow Image 6
%
hr=zeros(250); hg=zeros(250); hb=zeros(250);
for i=1:m2
```

```
    for j=1:n2
        k=R6(i,j)+1; hr(k)=hr(k)+1;
        k=G6(i,j)+1; hg(k)=hg(k)+1;
        k=B6(i,j)+1; hb(k)=hb(k)+1;
    end
end
for k=1:250
   % hr(k)=100.0*hr(k)/(m2*n2);
   % hg(k)=100.0*hg(k)/(m2*n2);
   % hb(k)=100.0*hb(k)/(m2*n2);
end
% subplot(3,3,8)
% L=0:249;
% plot(L,hr,'r-',L,hg,'g-',L,hb,'b-')
% xlabel('Histogram of Shadow 6')
% ('No. of Pixels')
%
% (8) IC holds a perfect revealed image
%
for i=1:m
    for j=1:n
            R0(i,j)=R0(i,j)+mod(R(i,j),p);
            G0(i,j)=G0(i,j)+mod(G(i,j),p);
            B0(i,j)=B0(i,j)+mod(B(i,j),p);
    end
end
IC(:,:,1)=R0; IC(:,:,2)=G0; IC(:,:,3)=B0; imwrite(IC,'tRecov.tif');
subplot(2,4,8)
imshow(IC)
xlabel('Recovering by Shadows 1\sim4')
```

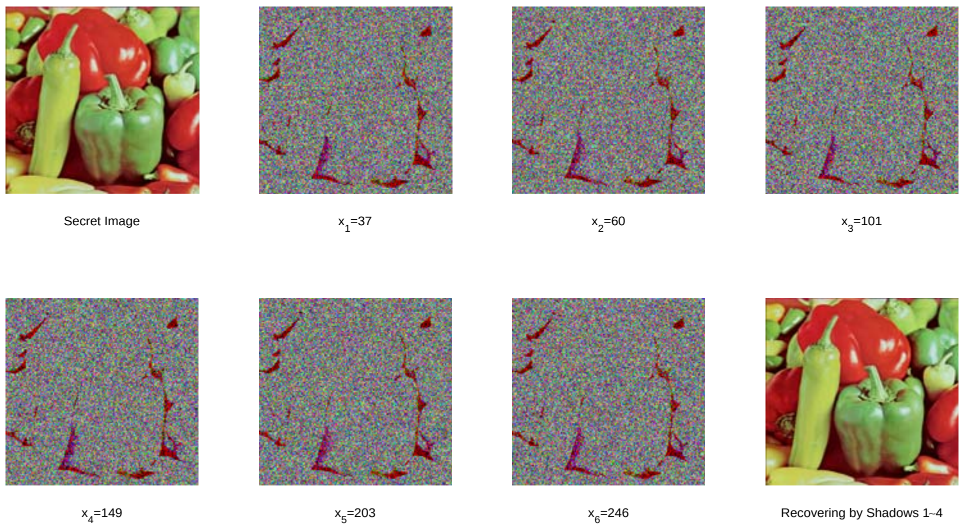

Figure 2: Shamir Shadows and Reconstruction for Peppers.