A Parallel Implementation of Multiple Secrete Image Sharing Based on Celullar Automata with LSB Steganography

Adrian Hernandez-Becerril[**], Mariko-Nakano-Miyatake[**],
Marco Ramirez-Tachiquin[*], Hector Perez-Meana[**]
[*] Universidad La Salle, Mexico City, Mexico
[**] National Polytechnic Institute/ESIME-Culhuacan, Mexico City, Mexico
adrian.hernandez.becerril@hotmail.com, mnakano@ipn.mx, marco.ramirez@lasallistas.org.mx, hmperezm@ipn.mx

***Abstract*— In this work we propose a parallel implementation of a Cellular Automata (CA)-based multiple secret image sharing (SIS) with a Least Significant Bit (LSB) Stenography. In previous CA-based multiple-SIS schemes, the resulting encrypted files are noise-like appearance images, which arises suspicion of observers when these are transmitted through a public network. To solve this problem, the LSB steganography is used to hide the encrypted noise-like images into camouflage images. To reduce execution time of the encoding and the decoding process of the SIS scheme, a parallel implementation based on CUDA technology is proposed. The simulation results demonstrate that the proposed parallel algorithm is more than 7 times faster than the conventional sequential algorithm. This reduction of temporal complexity and obtaining unnoticed shares allow the reliability of this scheme to be used in many information security fields such as the user-devices and cloud storage environments.**

I. INTRODUCTION

In 1979 Shamir and Blakley proposed secret sharing schemes [1, 2], which are cryptographic procedures to share a numerical secret key among a set of participants. These schemes have the particularity that only some qualified subset of the participants could recover the secret key. The scheme proposed by Shamir is based on the polynomial interpolation, while Blakely's approach is based on the intersections of some high dimensional planes in a high dimensional space.

In recent works the secret sharing schemes [1, 2] are applied to share secret images instead of numerical secret keys, this type of secret sharing schemes receive the name of secret image sharing (SIS) [3, 4]. The schemes based on secret image sharing are classified into two important categories. The first category includes the schemes based on the interpolation function (IF), specifically employing the Langrage's interpolation function [3, 5] for sharing the secret images. The second one consists of the schemes based on cellular automata (CA), in which the secret images are shared using local transition functions applied to a set of cells [4, 6]. The schemes in each category present advantages and disadvantages respect to its counterpart.

The main advantage of CA-based method is that multiple secret images can be encrypted while IF-based method only one secret image can be encrypted. In the other hand the main advantage of IF-based method is the flexibility by being a (k, n)-threshold encryption, where up to $k \leq n$ shares are required to reveal the secret, i.e. all participants shares are not required to reveal the secret, while CA-based method is a (n, n)-threshold encryption, in which all participants must provide their shares to reveal the secret by forcing to have all the participants shares to reveal the secret.

The main disadvantage of both categories of the SIS-schemes is the high computational complexity that avoids the practical use in commercial applications. In [7, 8], the authors proposed a parallel implementation of IF-based algorithms. Recently the authors of [9] proposed a scheme to handle the noise-like appearance shares by embedding them into cover images to construct stego-images. They proposed a reversible steganography method which distorted stego-image is reverted to the original one. In a practical use, stego-images are any images that are only used to hide secret information, so the reconstruction of the original cover images may be meaningless. Other disadvantages presented in [9] are an overflow issue and the maximal pixel value of the cover image is limited by different values of k of (k, n)-threshold scheme. Recently in [10] a parallel implementation of CA-based multiple-SIS scheme was proposed using the technology of CUDA, with the disadvantage that the secret image is a noise-like share.

In this paper we proposed a parallel implementation of CA-based multiple-SIS scheme [6] combined with a parallel implementation of the LSB stenography algorithm applying CUDA technology to form parallel structure [11]. This technology was built specifically for supercomputing applications as a parallel computing platform and programming model. The proposed implementation is approximately 7 times faster than the conventional sequential approach [6] in both encoding and decoding stages, keeping high quality of stego-images with the average Peak Signal to Ration (PSNR) of 43.5 dB.

978-1-4799-0421-1/13/$31.00 ©2013 IEEE

The rest of this paper is organized as follows: In Section 2, Multiple-SIS algorithm is described. In Section 3 the proposed parallel implementation of CA-based multiple-SIS is described together with experimental results and finally we conclude this work mentioning the advantage respect to the previous works.

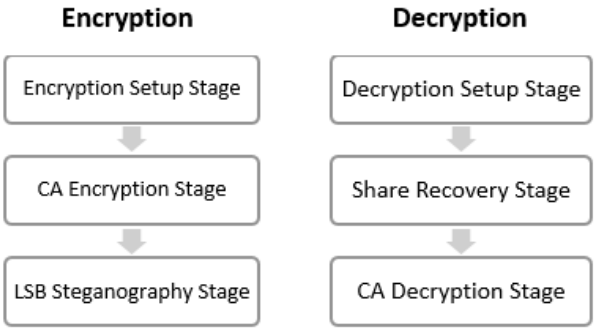

Figure 1 Diagram of the multi-secret image sharing scheme

II. MULTISECRETE SHEARING SCHEME

The multisecret sharing scheme that we treat is (n,n)-threshold scheme, in which each one of n participant $P_1,..,P_n$ shares n secret color images, $SI_1,...,SI_n$, among them. The secret images are encoded to generate n noise-like share patterns, which are hidden into n camouflage images $IC_1,...,IC_n$ using LSB steganography. The camouflage images are distributed among n participants, and all participants can recover all secret images if all of them provide their camouflage images $IC_1,...,IC_n$ to the dealer D. The principal reason of use of the LSB steganography in our scheme is its parallel implementation possibility. Even though LSB steganography is considered as an insecure scheme, in our case we embed a noise-like share instead of meaningful information and to recover the secret images, all shares must be extracted and perform CA-based SIS.
The secret image files used in this work have a Portable Network Graphics (PNG) format, this format supports palette based images of 32 bits in RGBA color space. Although the proposed scheme is designed to share n secret images with PNG format, any binary files, such as images files with any other formats, document files and also executable files, can handle as secret data. The camouflage images we used also have PNG format, due to its lossless compression property.

The proposed multiple-SIS scheme is divided in the following two phases: encryption and decryption phases, as shown by Fig. 1. Each phase is described in detail.

A. Encription Setup Stage

The dealer D defines the Linear Memory Cellular Automata (LMCA) with an order $n+1$ and its initial configuration as follows:

1. D receives from $P_1,...,P_n$ participants their secret images $SI_1,...,SI_n$ in PNG file format. Padding process of white pixels is required if the images have different sizes. Each image is represented as a matrix of $4\times$ *(width* $\times$ *height)* in the RGBA color space.
2. n random values $\omega_m \in [0,511]$, $m=1,...,n$ are generated in order to define m local transition functions f_{ω_m} of n linear Cellular Automata (LCA). The functions are represented as an eight bit value array.
3. The local transition function of the LMCA of the order $n+1$, $a_{i,j}^{(t+1)} = F(N_{i,j}^{(t)},...,N_{i,j}^{(t-n)})$ is constructed as follows:

$$F(N_{i,j}^{(t)},...,N_{i,j}^{(t-n)}) = \sum_{m=1}^{n} f_{\omega_m}(N_{i,j}^{(t-m+1)}) + a_{i,j}^{(t-m)} (\mathrm{mod}\, 256) \quad (1)$$

where $N_{i,j}^{(t)}$ is a neighborhood of the (i, j)-th cell in time t, which is composed 9 cells centered by (i, j)-th cell $a_{i,j}^{(t)}$.

4. D defines components of the intial configuration of LMCA, $C^{(m)} = SI_m$, $m = 1,...,n$ which correspond to the n secret color images.
5. D generates a color image SI_0 with random values, which has the same size and the same color palette as the other secret images. Then the initial configuration in time t=0, $C^{(0)} = SI_0$. A good random generator must be used to assure the security of the proposed scheme.

B. CA Encription Stage

The dealer D encrypts the secret images using CA-based SIS to generate the shadow images. The CA encryption stage is described as follows:

1. A positive integer value l is selected considering that $l \geq n+2$. The evolution of the LMCA defined in the setup stage is performed $(l+n-1)$ times from the initial configuration as shown by (2)

$$\{C^{(1)}, C^{(2)},..., C^{(n)}, C^{(n+1)},..., C^{(l-1)}, C^{(l)}, C^{(l+1)},..., C^{(l+n-1)}\} \quad (2)$$

2. Each configuration, $C^{(t)}, t > n$ of the evolution has a noise-like appearance. The n shadow $R_m = C^{(l+m-1)}$, $m=1,...,n$ correspond to the last n configurations of the evolution of the LMCA.

The evolution number and the last component of the initial configuration for the inverse CA, i.e., $R_0 = C^{(l-1)} = C^{(n+1)}$ are public for all participants.

C. LSB Steganography Stage

The dealer D hides each shadow R_m into the corresponding camouflage image, which are the following steps:

1. D receives n camouflage images $IC_1,...,IC_n$.
2. Each shadow R_m, m=1..n, together with image size (width and height) is hidden into each camouflage image IC_m.
3. Each is interpreted *(i,j)-th* pixel with 8 bit value. Three least significant bits of the pixels of each camouflage image $IC_{m_{i,j}}$ are used for data hiding.
4. Camouflage image with the hidden data are compressed by using the PNG file format.
5. D distributes the shares composed by three elements: (m, ω_m, IC_m), $m=1,...,n$, to each participant $P_1,...,P_n$, where m is the participant number, ω_m is the rule number and IC_m the assigned camouflage image to the m-th participant.

The global process of the encryption phase is shown by Fig. 2.

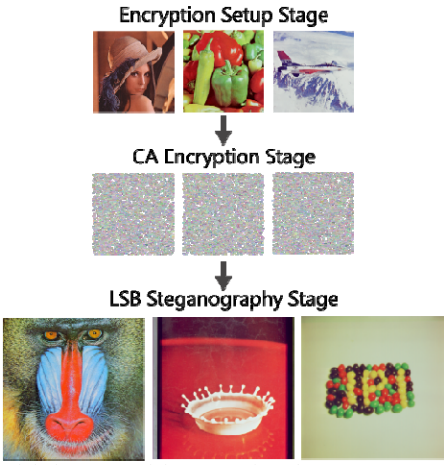

Figure. 2. Global process of the encryption phase

D. Decription Setup Stage

In this stage each participant provides their share (m, ω_m, IC_m), $1 \le m \le n$ to the dealer D. Additionally D knows l and R_0 because these parameters are public.

E. Shares Recovery Stage

The dealer D recovers each R_m from the received camouflage images $IC_1,...,IC_n$ performing the following steps:

1. Firstly, the dealer D recovers the size (width × height) of each shadow R_m, which was embedded in the initial positions of the camouflage image IC_m.
2. The three least significant bits of each pixel of the camouflage image IC_m are extracted to form 3 bits of R_m.

F. CA Decription Stage

The dealer D recovers the secret images using CA-based SIS method applying inverse evolution, which is as follows:

1. D recovers the secret color images applying inverse LMCA using R_m and ω_m provided from each of n participants. The inverse evolution of the LMCA is given by

$$\breve{C}^{(i)} = R_n = C^{(l+n-1)},..., \breve{C}^{(n)} = R_1 = C^{(n+1)} \\ ,..., \breve{C}^{(n+1)} = R_0 = C^{(l-1)} \quad (3)$$

The inverse evolution of the LMCA is iterated *l-1* times applying the local transition function, $a_{i,j}^{(t+1)} = G(N_{i,j}^{t},..., N_{i,j}^{(t-n)})$, which is given by

$$G(N_{i,j}^{(t)},..., N_{i,j}^{(t-n)}) = -\sum_{m=1}^{n} f_{\omega_m}(N_{i,j}^{(t-m+1)}) + \\ a_{i,j}^{(t-n)} (\mathrm{mod}\, 256), \quad (4)$$

where $N_{i,j}^{(t)}$ is a neighborhood of the (i, j)-th cell in time *t*.

2. The dealer D obtains the secret color images performing (5).

$$\breve{C}^{(l)} = C^{(n)} = a_n \breve{C}^{(l+1)} = C^{(n-1)} = a_{n-1,...,} \\ \breve{C}^{(n+l+1)} = C^{(1)} = a_1 \quad (5)$$

The global process of the decryption phase is shown by Fig. 3.

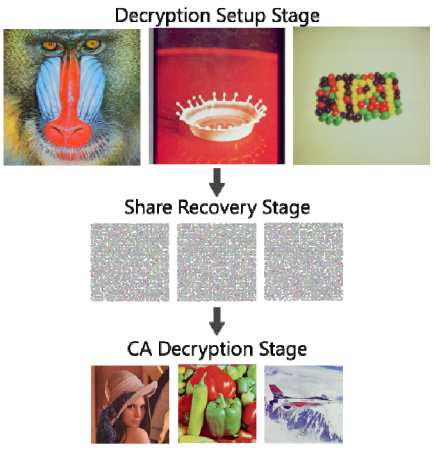

Figure. 3. Global process of the decryption phase

III. Proposed Parallel Implementation

The proposed parallel implementation using the CUDA technology reduces considerably the temporal complexity of the sequential implementation of the multiple-SIS scheme. Additionally the proposed scheme hides the

shadow images R_m, generated by the CA-SIS, into the camouflage images by applying a parallel implementation of the LSB stenography algorithm.

The main steps for the parallel implementation of encrypting are following:

1. Receive n secret images $SI_1,...,SI_n$ and n camouflage images $IC_1,...,IC_n$.
2. Create randomly C_0 and $\omega_1,...,\omega_n$.
3. Copy the four color values (R, G, B, A) of each pixel of the secret images $SI_1,...,SI_n$ and n to an array with the size $(n+1)\times height\times width\times 4$.
4. Copy the four color values (R, G, B, A) of each pixel of the camouflage images $IC_1,...,IC_n$ to an array with the size $(n)\times height\times width\times 4$.
5. In this work we consider that $l=2(n+1)$. The processing time of the parallel solution with the conventional sequential one is performed using same l in the encryption and decryption process.
6. Allocate GPU buffers and copy host memory to GPU buffers for C_0, $SI_1,...,SI_n$ and $IC_1,...,IC_n$. In this work we use (8,8)-threads scheme per block and the total number of blocks N_b is defined as

$$N_b=\frac{width}{thread\ \ perblock}\times\frac{height}{thread\ \ perblock} \qquad (6)$$

   We make a memory copy between host and device memory only two times to minimize the execution time (load-download), because each copy of memory implies a penalty time. All secret images and camouflage images are loaded in the device memory. The intermediate $C^{(t)}$ are not saved in the process, reducing the required memory available in the device.
7. Compute in parallel each pixel value of each color (RGBA) from *(l-1)* to *(l+n)* getting a new C in the process. Here to realize the evolution in parallel form, D sends each cell value *(pixel value)* in a separate thread in the GPU, having *width* × *height* threads.
8. The n shadows $R_m=C^{(l+m-1)}$, $m=1,...,n$ are the last n configurations of the evolution of the LMCA. Next, we compute in parallel each pixel value (RGBA) fro R_m, hiding it in each IC_m. To perform this process in parallel form, D sends each cell value (pixel value) in a separate thread in the GPU, having *width x height* threads. The three least significant bits are considered in this process,
9. Each $IC_1,...,\ IC_n$ is compressed with the PNG file format.
10. D distributes the shares (m,ω,IC_m) to each participant.

The main steps of the parallel decryption process are described as follows:

1. Receive each share (m,ω,IC_m) from each participant and also public parameters l and R.
2. Recover size *(width* × *height)* of each shadow R_m from each camouflage image IC_m.
3. Declare the required memory for the four color values of each pixel (R, G, B, A) to an array with the size $(n+1)\times height\times width\times 4$ for the secret images $SI_1,...,SI_n$ and C_0.
4. Copy the four color values of each pixel (R,G,B,A) of the camouflage images $IC_1,...,IC_n$ to an array with the size $(n)\times height\times width\times 4$.
5. Allocate GPU buffers and copy host memory to GPU buffers for C_0, $SI_1,...,SI_n$ and $IC_1,...,IC_n$. In this work we consider the (8,8)-th scheme per block and the total number of blocks is given by (6). We make a memory copy between host and device memory only two times to minimize the execution time (load-download). All secret images and camouflage images are loaded in the device memory. The intermediate $C^{(t)}$ are not saved in the process, reducing the required memory available in the device.
6. Recover each R_m from IC_m. To perform this process in parallel form, D sends each cell value (pixel value) in a separate thread in the GPU, having *weight* × *height* threads. The three least significant bits are considered in this process.
7. To compute the inverse evolution of the LMCA in parallel form, D sends each cell value (pixel value) of time step t in a separate thread in the GPU, having *weight* × *height* threads. Compute in parallel each pixel value from *l-1* to *l+n* getting a new C in the process.
8. Finally, save the recovered secret images.

IV. Evaluation of Proposed Scheme

To evaluate the proposed parallel implementation of the SIS scheme, three important measures were considered, which are temporal complexity, quality and size of the camouflage images. The number of participants, n, is varied from 3 to 5, and four different sizes of images, 1280×800, 1440 ×900, 1680×1050 and 1920×1200 pixels are considered. Table I shows the executive time in the sequential encryption implementation, while Table II shows that in the proposed parallel implementation under the above mentioned condition. Tables III and IV show the executive time to perform the decryption process, in the sequential and parallel implementations, respectively. The time is measured in seconds and only considering the computation of C from *l-1* to *l+n*, no I/O time consuming is taken into account. The parallel processing time reported in Tables II and IV, includes the penalty time due to transferring the host memory to device memory.

TABLE I. Encryption-sequential implementation

n	1280x1440	1440x900	1680x1050	1920x1200
3	5.50s	6.93s	9.47s	12.27s
4	7.53s	9.51s	12.91s	16.84s
5	9.82s	12.38s	16.74s	21.98s

TABLE II. ENCRYPTION-PROPOSED PARALLEL IMPLEMENTATION

n	1280x1440	1440x900	1680x1050	1920x1200
3	0.81s	0.99s	1.33s	1.67s
4	1.15s	1.40s	1.88s	2.35s
5	1.51s	1.85s	2.52s	3.14s

TABLE III. DECRYPTION-SEQUENTIAL IMPLEMENTATION

n	1280x1440	1440x900	1680x1050	1920x1200
3	6.19s	7.87s	10.67s	13.97s
4	8.37s	10.62s	14.44s	18.87s
5	10.87s	13.73s	18.80s	24.42s

TABLE IV. DECRYPTION-PROPOSED PARALLEL IMPLEMENTATION

n	1280x1440	1440x900	1680x1050	1920x1200
3	0.82s	0.99s	1.33s	1.67s
4	1.14s	1.39s	1.87s	2.35s
5	1.51s	1.85s	2.52s	3.15s

The hardware used to obtain these results is a GeForce 555M and i7 Intel processor at 2.00 Ghz. This GPU has 144 cores and DDR3 type memory; better results could be obtained with a faster GPU with more cores as the GPU GeForce GTX 680M or TESLA GPUs. In the proposed scheme two performance factors are important 1) the number of processing cores, because each pixel is processed in a parallel thread, not all threads could be processed simultaneously, which depends on the cores of the GPU so with more cores more pixels could be processed in parallel, reducing the execution time. And 2) memory performance, faster memory allows bigger chunks of bytes being transferred without losing performance. The results presented in this work could be improved significantly in the next years with faster GPUs and with the increased of processing cores. The posed parallel implementation is approximately 7 times faster than the conventional sequential implementation in both encryption and decryption processes.

Table V shows the average PSNR of the camouflage images respect to their original one. The size of the camouflage images is 5184×3240 pixels and their file size is 5.18 MB in a PNG file format. Table VI shows the average size in MB of the camouflage images. The use of the PNG file format allows a smaller size of the camouflage images due to its lossless compression property.

TABLE V. AVERAGE PSNR OF CAMOUFLAGE IMAGES

n	1280x1440	1440x900	1680x1050	1920x1200
3	45.24 dB	44.21 dB	42.88 dB	41.71 dB
4	45.24 dB	44.21 dB	42.88 dB	41.71 dB
5	45.24 dB	44.21 dB	42.88 dB	41.71 dB

TABLE VI. AVERAGE SIZE OF CAMOUFLAGE IMAGES

n	1280x1440	1440x900	1680x1050	1920x1200
3	9.57 MB	10.8 MB	13.1 MB	15.6 MB
4	9.57 MB	10.8 MB	13.1 MB	15.6 MB
5	9.57 MB	10.8 MB	13.1 MB	15.6 MB

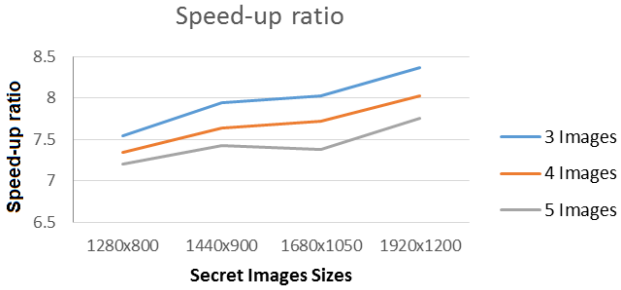

Figure. 4. Speed-up ratio of the parallel implementation with respect to the sequential implementation

Fig. 4 shows the speed-up ratio of the parallel implementation respect to the sequential one, varying size of the secret image. This figure also shows relationship between this ratio and the number of the secret images (number of participants). The results show that the performance of the proposed parallel implementation increases according to the image size, when the number of the secret images is same. The speed-up ratio decrease when the number of the secret images increases. The reason of this behavior is because more memory must be transferred from the host memory to the device memory which is a time consuming task.

V. CONCLUSIONS

The main goal in this work was to implement the least significant bit (LSB) stenography algorithm to hide the shares into camouflage images and the parallel implementation to reduce execution time of the proposed CA-based SIS. Two main advantages are obtained 1) the average reduction of execution time due to parallel implementation is approximately 7 times faster compared with the execution time of the conventional sequential implementation [6] 2) Hiding the share into camouflage images using LSB steganography avoid arising suspicious of third party. Users could have these encrypted files in their mobile devices in the photographs album with encrypted data passing unnoticed and only the user could identify the encrypted files. There is a tendency in the mobile operating systems to store all information from mobile devices in cloud services, even though it is a good solution for information storage, it weakens the security of the stored information. This scheme offers a good solution for encrypting the information and if an attack to the server is done there is no easy way to identify the files as encrypted and they cannot recover the information due to a (n, n)-threshold scheme.

ACKNOWLEDGMENT

We thank the National Science and Technology Concil of Mexico for the financial support during the realization of this research.

REFERENCES

[1] A. Shamir, How to share a secret, *Communication ACM* , Vol.22, 1979, pp. 612-613.

[2] G. Blakley, Safeguarding cryptographic keys, National *Conf. on AFIPS*, Vol. 48, 1979, pp. 313-317.

[3] C-C. Lin, W-H. Tsai, Secret image sharing with steganography and authentication, *Journal of systems and Software*, Vol. 73, 2004, pp. 405-414.

[4] J. Jin, Z-H. Wu, A secret image sharing based on neighborhood configurations od 2-D cellular automata, *Optics & Laser Technology*, Vol. 44, 2012, pp. 538-548.

[5] C.-C. Chang, Y.-P- Hsieh, C.-H. Lin, Sharing secrets in stego images with authentication, *Pattern Recognition,* Vol. 41, 2008, pp. 3130-3137.

[6] G. Alvarez, L. H. Encinas, A. Martín del Rey, A multisecret sharing scheme for color images based on cellular automata, *Information sciences*, Vol. 178, 2008, pp. 4382-4395.

[7] W. P. Fang, Parallel Processing for Secret Image Sharing, *Int. Symp. on Parallel and Distributed Processing with Applications,* Vol. 1, 2010, pp. 392-396.

[8] W. P. Fang, S. J. Lin, Fast Secret Image Sharing scheme in HPC, *Int. Conf. on High-Performance Computing*, 2009.

[9] Wu, X., et al., A user-friendly secret image sharing scheme with reversible stenography based on cellular automata, J. Sys. Software (2012), doi: 10.1016/j.jss.2012.02.046.

[10] R. A. Hernandez-Becerril, M. Nakano-Miyatake, M. Ramirez-Tachiquin, H. Perez-Meana, A Parallel implementation of multiple secret image sharing based on cellular automata, *Advances in Circuits, Systems, Automation and Mechanics*, 2012, pp. 33-38.

[11] M. Garland, Parallel computing with CUDA, *Parallel amp Distributed Processing IPDPS 2010 IEEE International Symposium,* 2010.