A Parallel Authenticated Encryption Sharing Scheme Based on Cellular Automata

Adrian Hernandez-Becerril, Mariko Nakano-Miyatake, Hector Perez-Meana,
A. Bucio R., *Member, IAENG* and M. P. Ramirez -Tachiquin, *Member, IAENG*

***Abstract*— Multi-secret sharing scheme based on cellular automata have proven to be a secure encrypting algorithm, although it cannot guarantee data integrity and authenticity of shares of the participants, allowing a chosen cipher text attack. In this work, to improve the security of multiple secret sharing scheme (SSS) against adaptive chosen cipher text attack, we introduce an authenticated encryption (Encrypt-then-MAC) based on Keccak function to a cellular automata based SSS. Taking the advantage that each cell of the cellular automata can be processed independently, we parallelize the processes of the SSS through CUDA technology, obtaining considerable reduction of the temporal complexity. The processing time of the proposed scheme is evaluated comparing with its sequential version and the speed-up rate of the parallel algorithm respect to the sequential one is 34 times in the best case.**

***Index Terms*— Authenticated Encryption, Cryptography, Cellular Automata, Parallel Processing, Secret Sharing**

I. Introduction

PRIVACY becomes an important concern, because recently we are uploading big amounts of personal data to the public cloud. Not only in personal devices, such as cellphones, tablets or personal computers, but also in new datacenters created by IaaS (infrastructure as a service) providers, our personal information is uploaded with different security characteristics and capabilities. The personal information may be vulnerable to different attacks and leaked to different entities. The vulnerabilities of these data arise not only in the infrastructure layer but also in the application layer. Recent reports indicate an increase of new ways of capture these personal data in the application layer, in order to be analyzed for different purposes, from marketing to spying. These applications could be disguised as games, web pages, email or even hosting providers.

During several years, new cryptographic algorithms with different characteristics and capabilities have been developed and implemented, since new attacks discover vulnerabilities and deficiencies of the conventional algorithms. In the public cloud applications, much information is shared among some authorized persons, in which the concept of secret sharing is required. The secret sharing schemes (SSS) were proposed separately by Shamir and Blakley in 1979 [1, 2]. Shamir introduced the method based on polynomial interpolation while Blakley proposed an approach to the secret by the intersections of some high dimensional planes in a high dimensional space. In the both cases, a numerical key is shared among some participants such that only subset of the qualified participants can recover the secret key. Recently the secret image sharing (SIS) schemes [3-7] are derived from SSS [1,2], in which secret images are shared instead of a numerical secret key. The SIS schemes are classifies in two categories: interpolation function (IF) based SIS, in which secret images are shared employing Lagrange interpolation function [3,4], and the cellular automata (CA) based SIS, in which secret images are shared using a local transition applied to a set of cells [5,7].

The main advantage of the CA-based method is that multiple secret images can be encrypted and shared, while the IF-based method only one secret image can be encrypted. This advantage of the CA-based method presents a management of a great amount of secret data, although it also presents a high computational complexity. Additionally the conventional CA-based SIS scheme can guarantee neither the integrity nor authenticity of the encrypted shares. Applying a chosen cipher text attack, which is widely studied attack in stream and block ciphers, the decrypted results can be modified.

In this work, we propose a CA-based secret sharing method with an authenticated encryption scheme (Encrypt-then-MAC) using a keyed-hash message authentication code based on Keccak function [16]. In the proposed scheme, any binary files, including image files can be used as secret data instead of secret images used in the conventional CA-based SIS [5], therefore the proposed scheme is a CA-based multiple secrets sharing scheme (Multi-SSS). In the proposed Multi-SSS, the integrity and authenticity for each of the encrypted shares is guaranteed by the Keccak function based MAC. In order to obtain high speed operations of the SSS, the CUDA technology is implemented to parallelize local transition operations of CA although the Keccak hash function is performed in batch mode. The evaluation of the processing time of proposed parallel Multi-SSS compared with its sequential version shows considerably speed-up, which is more than 34 times in a large number of participants.

The rest of this paper is organized as follows: In Section 2,

A Hernandez-Beccerril, Mariko Nakano-Miyatake, Hector Perez-Meana and M. P, Ramírez-Tachiquin: Postgraduate Section of Mechanical Engineering School, Instituto Politecnico Nacional Mexico City, Mexico.

A. Bucio R. : UPIITA-IPN, National Polytechnique Institute, Mexico, ari.bucio@gmail.com.

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

the Linear Memoryless CA algorithm, which is the base of proposed Multi-SSS. In Section 3, we described the proposed parallel implementation of proposed Multi-SSS. In Section 4, the experimental results are presented and finally the conclusions of this work are given in Section 5.

II. CELLULAR AUTOMATA

The Cellular automata (CA) are mathematical models in which each cell of CA evolves through a number of discrete time steps according to a set of rules based on state of the cell and its neighborhood [5, 10-13]. Two-dimensional CA are defined by a 4-uplet (*C, S, N, f*), where *C* is the cellular space defined by a finite two-dimensional array of $r \times c$ cells, *S* is state of each cell, *N* defines neighborhood and *f* is local transition function that indicates evolution manner of each cell. $a_{i,j}$ denotes (*i ,j*)-th cell in *C* and its state in time step *t* is denoted as $a_{i,j}^{(t)}$. The value of each $a_{i,j}^{(t)}$ belongs to *S*, where $S = \mathbb{Z}_2 = \{0,1\}$. The local transition function f in each discrete time step *t* is applied to each cell $a_{i,j}^{(t)}$, *i*=1..*r*, *j*=1..*c* of *C* using the neighborhood $N_{i,j}$ of $a_{i,j}$ to obtain the state of time step *t*+1, $a_{i,j}^{(t+1)}$. In the CA, the Moore neighborhoods are generally employed [5, 6] consisting of a center cell and its eight nearest cells around the center as shown by

$$N_{ij} = \{a_{i-1,j-1}, a_{i-1,j}, a_{i-1,j+1}, a_{i,j-1}, a_{i,j}, a_{i,j+1}, a_{i+1,j-1}, a_{i+1,j}, a_{i+1,j+1}\} \quad (1)$$

At time step $t+1$, the updated state of cell a_{ij} is given by

$$a_{i,j}^{(t+1)} = f\left(N_{i,j}^{(t)}\right) = f(a_{i-1,j-1}^{(t)}, \dots, a_{i,j}^{(t)}, \dots, a_{i+1,j+1}^{(t)}) \quad (2)$$

where $1 \le i \le r, 1 \le j \le c$ and $N_{i,j}^{(t)}$ is the set of states of the neighbor cells of $a_{i,j}$ at time t. Since the number of cells on *C* is finite, boundary conditions must be considered in order to assure the well-defined dynamics of the cellular automaton. In [5], periodic boundary conditions are taken, that is:

$i \equiv u \pmod r$ and $j \equiv v \pmod c$.

The local transition function *f* defines the property of CA, if f is lineal, CA becomes linear CA (LCA) and it has inverse transition function f^{-1}. The local transition function of a LCA is given by

$$a_{i,j}^{(t+1)} = f\left(N_{i,j}^{(t)}\right) = \sum_{\alpha,\beta\in\{-1,0,1\}} \lambda_{\alpha,\beta} a_{i+\alpha,j+\beta}^{(t)} \pmod 2 \quad (3)$$

where $1 \le i \le r, 1 \le j \le c$ and $\lambda_{\alpha,\beta} \in \{0,1\}$ is defined by a rule number ω, which is an integer selected from $[0, 2^{nn}-1]$, where *nn* is the number of neighbor cells of $a_{i,j}$. Here *nn* is equal to 9. The rule number ω is expressed by

$$\omega = \lambda_{-1,-1}2^8 + \lambda_{-1,0}2^7 + \lambda_{-1,1}2^6 + \lambda_{0,-1}2^5 + \lambda_{0,0}2^4 + \lambda_{0,1}2^3 + \lambda_{1,-1}2^2 + \lambda_{1,0}2^1 + \lambda_{1,1}2^0 \quad (4)$$

If the rule number ω is selected, $\lambda_{\alpha,\beta}, \alpha, \beta \in$ {-1,0,1} are obtained uniquely and the updated state of $a_{i,j}^{(t+1)}$ is also uniquely determined. The local transition function of the LCA with rule number ω is denoted as f_ω.

The CA is classified into memoryless CA and memory CA (MCA). In the memoryless CA, the updated state of a cell is just determined by its neighborhood at only preceding time step, while in MCA, the state at time $t+1$ depends on the state of neighboring cells at time *t* as well as $t-1, t-2, \dots$, [5 , 10-13]. In this work, we consider *p*-th order linear MCA (LMCA) whose local transition function is given by

$$a_{i,j}^{(t+1)} = F\left(N_{i,j}^{(t)}, \dots, N_{i,j}^{(t-p+1)}\right) (mod 2) = f_{\omega_1}(N_{i,j}^{(t)}) + \cdots + f_{\omega_p}(N_{i,j}^{(t-p+1)}) (mod\ 2) \quad (5)$$

where $1 \le i \le r, 1 \le j \le c$ and $\omega_1, \dots, \omega_p \in [0, 511]$. In this case, in order to start the evolution of the LMCA, p initial configurations $\{C^{(0)}, \dots, C^{(p-1)}\}$ are required. The LMCA defined by (5) is reversible and its reserve transition function is given by

$$a_{i,j}^{(t+1)} = G\left(N_{i,j}^{(t)}, \dots, N_{i,j}^{(t-p+1)}\right) (mod\ 2) = f_{\omega_{p-1}}\left(N_{i,j}^{(t)}\right) - \dots - f_{\omega_1}\left(N_{i,j}^{(t-p+2)}\right) + a_{i,j}^{(t-p+1)} (mod\ 2) \quad (6)$$

where $1 \le i \le r, 1 \le j \le c$ *and* $\omega_1, \dots, \omega_p \in [0,511]$.

III. PROPOSED PARALLEL AUTHENTICATED ENCRYPTION SSS

The proposed Multi-SSS is a (*n ,n*)-threshold scheme, in which *n* participants share *n* secret files $SF_1, \dots, SF_n$ among them and to reveal these *n* secret files, all participants must provide their shares. In the scheme a dealer *D* controls all processes, managing several secret keys. To perform parallel implementation of the proposed scheme, each secret file is represented by means of a byte cuboid C_m, *m=1,…,n*, of size $x \times y \times z$, which stands for the configuration of neighborhood of the CA.

The dealer *D* randomly defines the values of x, y and z of the cuboid size, which are stored as secret keys. Additional some random bytes are added depending in the difference between $(original\ file\ size_m + header\ size)$ and $(x * y * z)$. Different combination of (*x, y, z*) gives different neighborhood configurations and as consequence of this, different cipher texts CT_m, m=1..n, are obtained as a part of shares of each participant.

The proposed Multi-SSS scheme is divided in the three phases: Setup phase, Sharing phase and Recovery phase [5]. Each of them is described in detail.

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

A. Setup Phase

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

- *D* receives each $SF_1, \dots, SF_n$, and generates a set of *n* cuboids based on the (x, y, z) values.
- *D* generates *n* random integer numbers $\omega_m \in [0{,}511], m = 1, \dots, n$ for each plane (x, y) in a value *z,* in order to generate local transition function f_{ω_m} .
- *D* constructs the local transition function of the LMCA of the order *n+1*, which is

$$a_{i,j}^{(t+1)} = F\left(N_{i,j}^{(t)}, \dots, N_{i,j}^{(t-n)}\right) \quad (7)$$

where

$$F(N_{i,j}^{(t)},..,N_{i,j}^{(t-n)}) = \sum_{m=1}^{n} f_{\omega_m}(N_{i,j}^{(t-m+1)}) + a_{i,j}^{(t-n)} \pmod{256} \quad (8)$$

- *D* defines *n* components of the initial configuration of LMCA: $C^{(m)} = C_m$, with $m = 1, \dots, n$, which are the n cuboids generated from *n* secret files.
- *D* generates a random cuboid based on True Random Number Generators (TRNGs), the security of this implementation relies in the generation of random values. In this work we are not using Pseudo-Random Number Generators (PRNGs) since most of the common libraries could not be sufficient for cryptography purposes and this type of random value generators are not suitable for security algorithms. In this work we use TRNGs in which the randomness is extracted from physical phenomena, such as radioactive source decays or atmospheric noise. These byte files are consumed by the encryption process to generate array S_0 with the size $x \times y \times x$, and considers $S_0 = C^{(0)}$, in order to complete the initial configuration of LMCA of order $n + 1$.

B. Sharing Phase

In this phase the dealer *D* computes in parallel the LMCA evolution for $C_0, \dots, C_n$. These operations are given as follows:

- *D* generates an integer number $l \geq n + 2$ in random form. This number determines the iteration of evolution.
- *D* evolves $(l + n - 1)$ times the LMCA defined in the setup phase. Here to realize the evolution in parallel form, *D* sends each cell value $za_{i,j}^{(t)}$ in an individual thread in the GPU, having $x \times y$ threads, the maximum number of threads depends on the GPU specification. It is worth noting that each cell value consists of *z* values is processed in each (x, y) thread. Through the evolution of the LMCA, we get matrices $\{C^{(0)}, C^{(1)}, \dots, C^{(n)}, \dots C^{(l-1)}, C^{(l)}, \dots, C^{(l+n-1)}\}$. Each configuration $C^{(t)}, t > n$, of the evolution has a noise-like appearance.
- *D* generates the shares $(m, z * \omega_m, CT_m), m = 1, \dots, n$, for each participant $P_1, \dots, P_n$. Each share is composed by three elements: the participant number m, its rule number $z * \omega_m$ and the cipher text CT_m. The *n* cipher texts $CT_m = C^{(l+m-1)}, m = 1, \dots, n$, are the last n configurations of the evolution of the LMCA.
- The iteration number of the evolution l and the last component of the initial configuration for the inverse CA, i.e., $R_0 = C^{(l-1)} = \tilde{C}^{(n+1)}$ are public for all participants.

C. Recovery Phase

In this phase, the dealer *D* receives the shares from all participants so that all of them can recover the *n* secret files. The operations of this phase are as follows:

- The dealer *D* will receive the share $(m, z * \omega_m, CT_m)$, $1 \leq m \leq n$ of each participant. Additionally *D* knows l and R_0 because these parameters are public.
- If all n participants are valid, the values (*x, y, z*) are obtained from the private key to construct the corresponding cuboid for each participant, so $(n + 1)$ cuboids $C_1, \dots, C_n$ of size $(x \times y \times z)$ are created. Here an incorrect size definition of each C will prevent an expected decryption result since the algorithm is sensitive to $(x \times y \times z)$ variables, different combinations of (x, y, z) values provide different plain text $SF_1, \dots, SF_n$.
- *D* reveals the secret files applying inverse LMCA using the initial configuration of the inverse CA provided by *n* participants and public random cuboid. The inverse evolution of the LMCA is iterated $l - 1$ times applying the local transition function, $a_{i,j}^{(t+1)} = G\left(N_{i,j}^{(t)}, \dots, N_{i,j}^{(t-n)}\right)$, which is given by

$$G\left(N_{i,j}^{(t)}, \dots, N_{i,j}^{(t-n)}\right) = -\sum_{m=1}^{n} f_{\omega_m}(N_{i,j}^{(t-m+1)}) + a_{i,j}^{(t-n)} (mod\ 256) \quad (9)$$

  To perform the inverse evolution of the LMCA in parallel form, reducing considerably computational complexity, *D* sends each cell value in a separate thread in the GPU, having $x \times y$ threads.

- Finally *D* obtains the n secret files: $SF_1, \dots, SF_n$.

The processing times for the sharing and the recovery phases are the same which consist in obtaining $l - 1$ in the evolution and inverse evolution of the given CA.

D. Authenticated Encryption

To authenticate cipher text $CT_1 \dots CT_n$ of each participant $P_1 \dots P_n$ and assure its integrity, we use a keyed-hash message authentication code (HMAC). The HMAC receives $CT_1 \dots CT_n$ and a secret key $K_1 \dots K_n$ known only by the participant and the dealer *D*. We generate $HCT_m = HMAC(K_m, CT_m)|CT_m, m = 1, \dots, n$ using key K_m and the cipher text CT_m. To validate each participant, the dealer *D* computes the MAC on the received CT_m using the same key K_m and HMAC function, and then compares the results with the received MAC. If the two values are same, CT_m has been correctly received, and the participant P_m is validated [17]. The length of each key K_m is 128 bytes.

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

HMAC requires a cryptographic hash function, denoted by H, and a secret key K stored by each participant P. In this work, Keccak, which is a family of hash functions based on the sponge construction, is considered as H. The sponge function is a generalization of the concept of cryptographic hash function with infinite output and works as authenticated encryption as well as pseudo-random generator. Considering the requirements of the proposed scheme and the desirable properties of Keccac, we selected the Keccac function as the hash function H for the HMAC.

The security of the message authentication mechanism depends directly on the cryptographic properties of the hash function: the resistance to collision and the message authentication property of the compression function

The definition of HMAC is as follows

$$HMAC(K,m) = H\left((K \otimes opad) \middle| H\left((K \otimes ipad)|m\right)\right)$$

HMAC is divided in two phases: encryption and decryption.

In the *HMAC* encryption phase, a keyed-hash message authentication code based on SHA-3 Keccak is used. Once the cipher texts obtained by CA sharing phase, the dealer D processes the Keccak based HMAC using cipher text CT_m and cryptographic key K_m of each participant P_m to obtain hash data HCT_m, m=1..n. The hash data $HCT_1, \dots, HCT_n$, which are used to verify the integrity and authentication of each cipher text before the recovery phase of CA, are assigned to each participant. In this work, the parameters used for Keccak function are $r = 1600, b = 1024$, respectively.

In the HMAC decryption phase, the dealer D receives from each of the n participants their cipher text and their hash values $HCT_1, \dots, HCT_n$. The dealer D validates each of the participants with their cryptographic key avoiding any cipher text chosen attack, ensuring data integrity and authentication for each participant.

E. Private Keys used in proposed Multi-SSS

In the proposed Multi-SSS, several private keys are managed to increase the security, which are as follows:

- l : The number of iterations required to evolution and inverse evolution of the CA, which equivalent to the encryption and decryption process.
- $(\omega_1, \cdots, \omega_n) * z$: The lambdas used in the local transition function of CA, which are determined by integer values ω and z.
- x, y and z: This values define the cuboid to be processed in parallel implementation of Multi-SSS.
- Cryptographic key for the HMAC of 128 bytes.

IV. Results

The number of processing cores of the GPU is a key factor of the processing speed in any parallel schemes. In the proposed scheme, z values in (x, y) are processed in a parallel thread, therefore more parallel threads implies less tasks to be performed with less time steps. In Table I, some new available GPUs are shown together with their specifications. In this work we used a GeForce GTX 770M due to its availability, however some newer GPUs with more processing cores, such as GeForce GTX 880M, can provide better performance from speed up points of view.

TABLE I
GPUs SPECIFICATIONS

	GeForce GTX 770M	***GeForce GTX 870M***	***GeForce GTX 780M***	***GeForce GTX 880M***
Chip	GK 106	GK 104	GK 104	GK 104
CUDA Cores	960	1344	1536	1536
Core clock base	811 Mhz	941 Mhz	823 Mhz	954 Mhz
Memory clock	2000 Mhz	2500 Mhz	2500 Mhz	2500 Mhz
Memory	3072 MB	6144 MB	4096 MB	8192 MB
Memory Interface	192 bit	192 bit	256 bit	256 bit

To evaluate time consuming of the proposed implementation of the CA-based Multi-SSS scheme in a parallel and sequential way, we vary the number of participants n={5,10,15,20,25}, and six different file sizes, 500 KB, 1000 KB, 2500 KB, 5000 KB, 10000 KB and 50000 KB. The total memory size in MB used in the proposed parallel implementation for different number of participants and total secret files size is described in Table II, which indicates the feasibility of proposed system in any GPU, although large number of participants share large amount of secret data.

TABLE II
TOTAL MEMORY USED IN EACH ENCRYPTION PROCESS IN MB.

n	***500 KB***	***1 MB***	***2.5 MB***	***5MB***	***10 MB***	***50 MB***
5	2.5	5	12.5	25	50	250
10	5	10	25	50	100	500
15	7.5	15	37.5	75	150	750
20	10	20	50	100	200	1000
25	12.5	25	62.5	125	250	1250

Fig. 1 shows the speed-up ratio between the parallel and sequential CA encryption implementation. In the actual available GPU, we achieved more than 34 times speed-up ratio compared with the conventional sequential implementation. Time is measured in milliseconds and only considering the computation of C from $l-1$ to $l+n$, no I/O time is taken into account. To compare the processing time of the parallel solution with the conventional sequential one in the encryption and decryption process in a fairly manner, the same $l = (n+2) * 2$ is used in both cases.

It is important to note that the same code is used in both, sequential and parallel CA encryption, leaving the compiler makes the adjustment to the two different hardware implementations. The difference of processing time between encryption and decryption can be negligible since only one arithmetic operation is different between them.

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

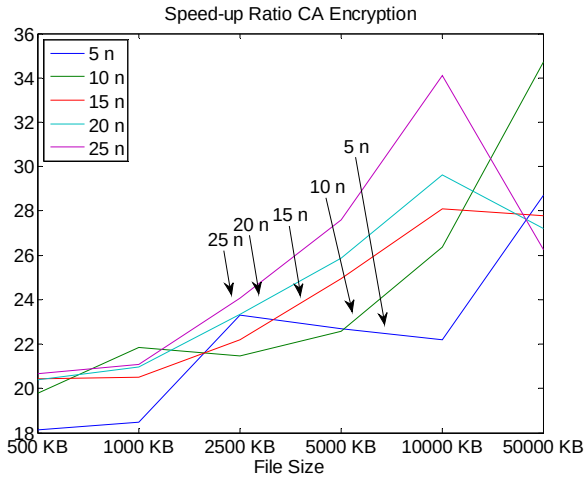

Fig 1. Speed-up Ratio between parallel and sequential implementation for CA Encryption.

Due to the Keccak hash function is not a natural parallel algorithm, the HMAC operation cannot be done in parallel, and instead we perform the Keccak algorithm in batch mode. As a future work, a parallel Keccak implementation is considered, decreasing moreover the total processing. So in this work, we only focused in the parallel Keccak batch mode, since in practical implementation, the dealer D can derive the process of CT_m of each participant P to the GPU. Fig. 2 shows speed-up ratio between GPU and sequential processor implementation under different number n of participants. It is worth noting that although the Keccak batch mode is used, we get more than 18 times of the speed-up ratio compared with the sequential operation if the number of participants is large, such as n=500. For a practical point of view for n <500, Keccak will be executed in sequential mode, CA speed-up ratio will be considered for algorithm comparisons between sequential and parallel implementations.

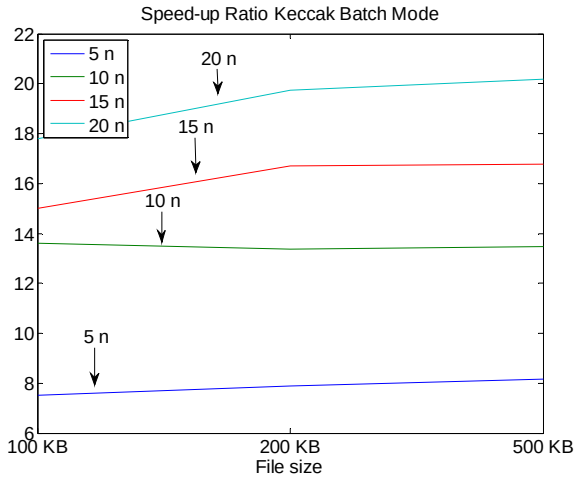

Fig 2. Speed-up Ratio between GPU and sequential processor implementation, in which HMAC is performed by Keccak batch mode.

As mentioned in Section III, the z array size is generated randomly to define the cuboid size, all z values will be processed by a single thread and its performance will depend on the z array size. In Fig. 3 we show different file sizes with different z array sizes under the number of participants $n = 5$, and from this figure we concluded that bigger z array sizes have a direct impact in the processing time in the GPU. The 5 times bigger z value provides 1.18 times more processing time compared with the smaller z array sizes. It is important to note that the size of the private key will depend on z, the smaller z provides smaller length of private key.

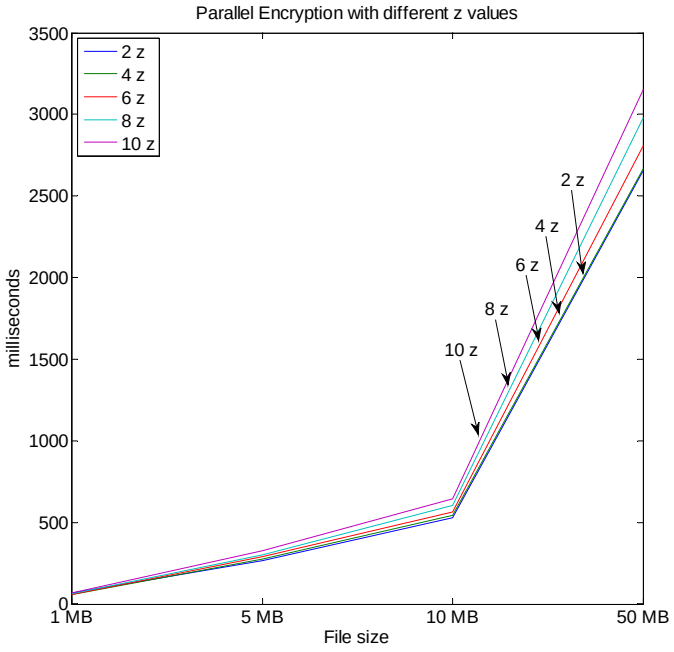

Fig 3. Time in milliseconds in the Parallel Encryption with different z values.

Fig. 4 shows an operation example of proposed Multi-SSS scheme using color images as the secret data. In this case, the number of participants is equal to 3 and they share three secret color images, as shown in Fig. 4(a), among them. The noise-like images obtained of LMCA after 10 time steps are shown in Fig. 4(b) and the recovered images after the inverse evolution of the LCMA are shown in Fig. 4(c). From the figure, we can observe that the proposed Multi-SSS performs correctly with an authentication mechanism of cipher texts (Fig. 4(b)) in the decryption process before inverse evolution of CA. The speed-up ratio of the parallel implementation respect to the sequential version in this particular example is approximately 22 times.

ISBN: 978-988-19252-0-6
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

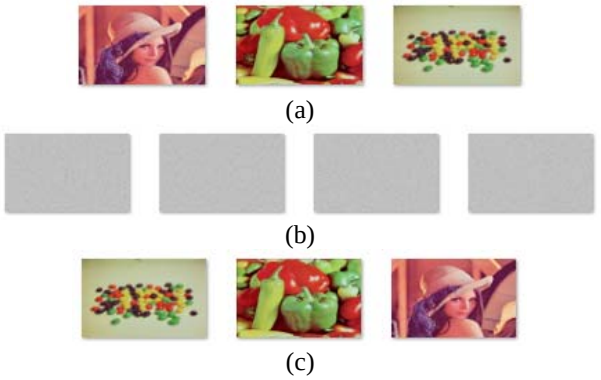

Fig. 4 (a) Secret images used in experiment, (b) shadows obtained the evolution of LMCA, (c) recovered secret image after the inverse evolution of LMCA.

V. Conclusions

The main purpose of this work was to propose a Multi-SSS that comply with high computation speed and mechanism of data integrity and authenticity using a keyed-hash message authentication code (HMAC) based on Keccak function. These characteristics are indispensable in new public cloud applications, in which big amounts of data are transferred and millions of users are interconnected sharing their information, which could be sensible data. To reduce high computational complexity required by the conventional CA-based Multi-SSS to manage a large amount of data, in proposed scheme, each secret file is converted in cuboid and operated in parallel using CUDA technology in GPUs. In the actual available GPU, we achieved more than 34 times speed-up ratio compared with the conventional sequential implementation. According to the hardware evolutions and emerge of new generation GPUs with higher speeds and higher number of cores, a higher reduction of temporal complexity of proposed scheme can be expected.

In proposed Multi-SSS, to guarantee data integrity and authenticity, an authenticated encryption algorithm in an encrypt-then-mac scheme and a keyed-hash message authentication code (HMAC) based on Keccak function is used. It is worth noting that in the speed-up ratio mentioned above, the whole process of the proposed Multi-SSS is considered, including HMAC operation, which must operate in batch mode due to lack of a parallel algorithm of Keccak function.

This authenticated encryption scheme is practically ideal for client-server applications in which the information must be shared with a group of users avoiding the server to recover the information without the explicit consent of all users.

As a future work, we consider the parallelization of the Keccak function to perform a complete parallelization of the Multi-SSS with authenticated encryption scheme, guaranteeing data integrity and authenticity in public cloud applications.

Acknowledgment

The authors thank CONACYT (Consejo Nacional de Ciencia y Tecnología/National Council for Science and Technology) for financial support received during this research.

References

[1] A. Shamir, How to share a secret, Communication ACM 22 (1979) 612-613.

[2] G. Blakley, Safeguarding cryptographic keys, in: National Conferenceon AFIPS, US, 1979.

[3] C.C. Lin, W.H. Tsai, Secret image sharing with steganography and authentication, Journal of Systems and Software 73 (2004) 405-414.

[4] C.C. Chang, Y.P. Hsieh, C.H. Lin, Sharing secrets in stego images with authentication, Pattern Recognition 41 (2008) 3130-3137.

[5] G. Alvarez, L.H. Encinas, A.M.D. Rey, A multisecret sharing scheme for color images based on cellular automata, Information Sciences 178 (2008) 4382-4395.

[6] J. Jin, Z.H. Wu, A secret image sharing based on neighborhood configurations on 2-D cellular automata, Optics & Laser Technology 44 (2012) 538-548.

[7] W.P. Fang, Parallel processing for secret image sharing, International Symposium on Parallel and Distributed Processing with Applications 1 (2010) 392-396.

[8] W.P. Fang, S.J. Lin, Fast secret image sharing scheme in HPC, in: International Conference on High-Performance Computing, India, 2009.

[9] M. Garland, Parallel computing with CUDA, in: Parallel AMP Distributed Processing IPDPS 2010 IEEE International Symposium 1 (2010) 1-2.

[10] N. Packard, S. Wolfram, Two-dimensional cellular automata, Journal of Statistical Physics 38 (1985) 901-946.

[11] T. Toffoli, N. Margolus, Invertible cellular automata: A review, Physica D 45 (1990) 229-253.

[12] R.A. Sanz, Reversible cellular automata with memory: two-dimensional patterns from a single seed, Physica D 175 (2003) 1-30.

[13] E. Fredkin, Digital mechanics, an informal process based on reversible universal cellular automata, Physica D 45 (1990) 254-270.

[14] G. Roelofs, PNG: The Definitive Guide, O'Reilly & Associates, Sebastopol, 2003.

[15] Adrian Hernandez-Becerril, Mariko Nakano-Miyatake, Marco Ramirez-Tachiquin, Hector Perez-Meana, Parallel Implementation of Multiple Secret Image Sharing Based on Cellular Automata, Journal of Communication and Computer 10 (2013) 649-660

[16] G. Bertoni, J. Daemen, M. Peeters and G. Van Assche, The Keccak reference, round 3 submission to NIST SHA-3, 2011

[17] Federal Information Processing Standards Publication, The Keyed-Hash Message Authentication Code (Hmac), http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf