IMAGE ENCRYPTION

Submitted by

Raghav Daruka

Megha Baid

Rishikesh S

160905472
160905480
160905105
CSE-A

Department of Computer Science & Engineering

MANIPAL
INSTITUTE OF TECHNOLOGY
A Constituent Institute of Manipal University, Manipal

Abstract

Matrix inversion is a significant in the GGH Encryption algorithm. The ability to invert large matrices quickly and accurately determines the effectiveness of a computational tool. Current literature suggests that time complexity of matrix inversion is 2 or higher. This paper redesigns the Gauss Jordan algorithm for matrix inversion on a CUDA and MPI platform to exploit the large scale parallelization feature of a massively multithreaded GPU. The Gauss Jordan method has been chosen for this project because it provides a direct method for obtaining inverse matrix and requires approx. 50% fewer operations unlike other methods. Hence forth it is suitable for massive parallelization. Further, the obtained result is used to decrypt the received encrypted image. The proposed method is simple and has a great potential to be applied to other situation where the exchange of images is done confidentially such as in military operation, banking transactions, message theory, medical reports etc.

Objectives

To parallelize the GGH Image encryption algorithm using MPI and CUDA and comparing the results with sequential implementation.

Introduction

Matrix inversion is an essential step in GGH Encryption. Until the late 1960s matrix inversion was believed to require a cubic number of operations, as the fastest algorithm known was Gaussian elimination method, or rather Gauss Jordan method which runs in O(n3) time where n is the size of the matrix. In 1969, Strassen excited the research community by giving the first sub cubic time algorithm for matrix multiplication, running in O (n2.808) time. This also reduced the time complexity, w, of Matrix Inversion using Strassen Multiplication to O(n2.808) time. This discovery triggered a long line of research that gradually reduced the time complexity w over time. In 1978, Pan presented a method that proved $w < 2.796$ and the next year, Bini et al. introduced the notion of border rank and obtained $w < 2.78$. Schönhage generalized this notion in 1981, proving $w < 2.548$. In the same paper, combining his work with ideas by Pan, he also showed $w < 2.522$. The following year, Romani found that $w < 2.517$. The first result to break 2.5 was by Coppersmith and Winograd who obtained $w < 2.496$. In 1988, Strassen introduced his laser method which was a novel approach for matrix multiplication, and thus decreased the bound to $w < 2.479$. Two years later, Coppersmith and Winograd combined Strassen's technique with a novel form of analysis based on large sets avoiding arithmetic progressions and obtained the famous bound of w

< 2.376 which has remained unchanged for more than 22 years (a very recent unpublished work claims to have brought down the limit to w < 2.3727). Recent developments in parallel architecture and its use in computation have brought about the prospect of massively parallel systems capable of reducing running times of codes below the limit of w = 2. The amount of performance boost of course depends largely upon the scope of parallelization that the algorithm provides. Owing to its unique architecture, the graphics processing unit (GPU) enables massive parallelization unlike anything possible on a CPU based network, as described later. The Gauss Jordan method is one of the oldest methods for matrix inversion. It is straightforward and robust and is particularly suitable for massive parallelization unlike many of the more advanced methods. Nevertheless, the available literature either on the scope of parallelization of the Gauss Jordan algorithm or its optimization appear rather insufficient. This paper tailors and implements the Gauss–Jordan algorithm for matrix inversion on a CUDA (Compute Unified Device Architecture) based GPU platform and studies the performance metrics of the algorithm.

Literature Review

The analysis on the Gauss elimination and matrix multiplication using CUDA and MPI environments have been carried out. The Gauss elimination method without pivoting was introduced to explain the concepts both in CUDA and MPI where in MPI, communication capabilities between the nodes increases apparently along with the floating point speed. The matrix multiplication equations have also been parallelized to further enhance performance and increase the algorithm efficiency. Pyhton language has been used to convert the image to a csv file (containing 2-d array with pixel values ranging from 0 to 255). These csv files are then taken as input in the C program which performs GGH encryption and decryption and produces the output image matrix in a csv file.

Methodology

Any lattice is a set of points in an n-dimensional space which has a periodic structure. Let

$$x_1, x_2, \ldots, x_n \in R^n$$

as n-linear independent vectors, the lattice generated by them is a vector defined as equation (1):

$$L(x1,x2,...,xn) = \{\sum i\} \qquad (1)$$

$x_1, x_2, \ldots, x_n \in R^n$ vectors are the basis of the lattice.

In this study, we have taken the advantage of the GGH algorithm for encrypting the clinical images. The GGH cryptosystem is based on CVP which is one of the NP-hard problems presented in 1997 by Goldreich et al. . It also introduces a trapdoor as a one-way function for implementing a public key cipher which relies on difficulty of lattice reduction. However, this algorithm was first cryptanalyzed by Phong Q. Nguyen in 1999. This system was a suggestive framework of the McEliece cryptosystem . For both systems, encryption is randomly performed. The basic GGH public key encryption system is similar to McEliece cryptosystem. The two parameters on which GGH relies, are lattice dimension ($n>200$) and the security parameter (σ). The security parameter presents the difficulty of the CVP. The authors of GGH, published some challenges for the security parameters as $n=200$, 250, 300, 350 and 400. Nguyen attacked all of the challenges except for $n=400$, since the key size was too large. The private key is a secret matrix R and its columns are comprised of a basis of a Lattice

$$L \subset Z^n$$

The parameters of GGH are shown in Table **1.**where the basis consists of the short integral vectors. There are several methods to construct the secret basis r. Two methods to generate the nice basis R are to choose a random matrix r within entries, (i.e., all vectors are chosen relatively short) and to choose $r = k.I_n+E$, where I_n is the $n\times n$ identity matrix, $k>1$is a medium sized integer, and E is a random matrix with small entries, as mentioned above. The public key is a public matrix denoted by B, which represents another basis for L. The public basis is known as a bad basis as it is not reducible as the secret basis. There are several methods which can randomly generate the public basis B from the secret basis r.

Table 1

GGH parameters.

Parameter	Description	Knowledge
n	Dimension	Public
σ	Security Parameter	Public
R	Integral matrix $n\times n$	Private
B	Integral matrix $n\times n$	Public

In 1999, Micciancio used Hermite Normal Form (HNF) for improving public key generation to reduce the size of the key. For generating the public key B from the private key R, a unimodular matrix U is required as shown in equation (2):

$$B = U.R \quad (2)$$

Now, assuming the message matrix as m and error matrix as e, the cipher matrix c is calculated as below:

$$c = m.B + e \quad (3)$$

To decrypt this cipher, the calculations are performed according to the following equations:

$$\text{round}(c.R^{-1}) = \text{round}((m.B+e)R^{-1}) = m.U.R.R^{-1} + \text{round}(e.R^{-1}) = m.U + \text{round}(e.R^{-1}) \quad (4)$$

The Babai rounding technique will be used to remove the term as it is a small value. Finally, the message matrix m will be calculated as follows:

$$m = m.U.U^{-1} \qquad (5)$$

If round $(e.R^{-1}) = b$ is a nonzero vector, the Rb will be a nonzero lattice vector. In this case, the Babai's rounding will not return the lattice point and hence, the wrong message will be retrieved. The decryption process will work correctly when round $(e.R^{-1}) = 0$. This will be possible when σ is small enough since the error vector is chosen from the vector $(\pm\sigma, \pm\sigma, \pm\sigma, \ldots, \pm\sigma)$. The increment of σ will increase the distance between the lattice vector m and the cipher text c. By increasing the distance, the CVP will become harder. When round$(R^{-1}e) \neq 0$ the probability of decryption error increases. The size of key and the complexity of this cipher are shown in Table **2.**

.

Table 2

Comparison of various parameters of standard and padding based GGH algorithms in terms of size of keys and complexity of key generation, encryption, and decryption.

Object	Size(bits)	
	GGH	Padding GGH
Private key R	$n2log2(k)$	$n2log2(k)$
Public key B	$n2log2(n)$	$n2log2(n)$
Operation	Complexity	
	GGH	Padding GGH
Key Generation	$n3$	$n3$
Encryption	$n2$	$2n2$
Decryption	$n2$	$2n2$

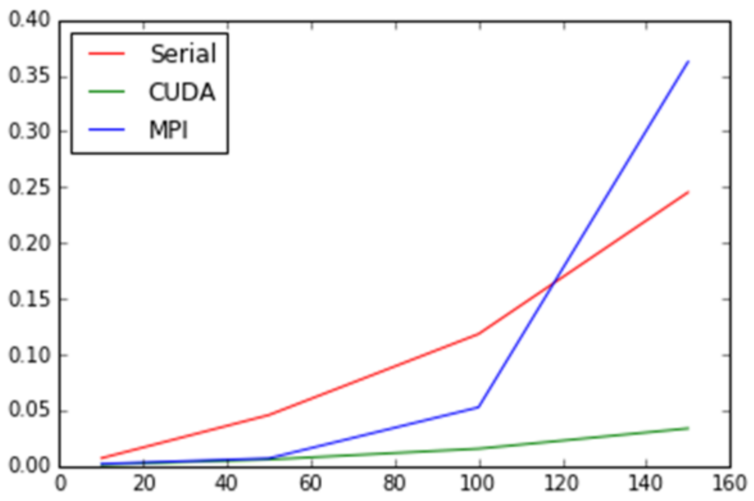
Suppose Bob is the receiver and Alice is the sender. He chooses an image, reduces the pixel values *mod 5*and takes this image as his private key. Then, he generates a unimodular matrix, randomly. He uses the equation (2) to calculate the public key. Then, Bob sends the public key image to Alice. Alice uses Bob's public key image to encrypt the clinical image or plain image using the equation (3). The cipher is a noisy image which is sent to Bob. Bob receives the cipher image and uses his own private key image and the unimodular matrix to decrypt the received image into an original clinical image using equations (4) and (5).

Steps

The GGH clinical cipher system has been implemented in C programming software. All the employed images are at the grey scale level. To encrypt a clinical image, at first, it has been resized into 200×200 pixels. Another image will be chosen and reduced by *mod 5* as the private key which will be read and resized to 200×200 pixels. A random matrix with size 200×200 where *det = 1, -1* will be generated randomly. Note that the size of 200×200 pixels is a sample size and any arbitrary size can be chosen for this step but it should be a square size and the larger the size, the more the lattice reduction will be difficult to be broken. According to the equation (2), the public key matrix is calculated. In this step, the private key and the unimodular matrix images are used for decryption process. The next step is to encrypt the clinical image using the public key image. The cipher image is produced according to the equation (3). After the image has been encrypted, the receiver receives the encrypted image, he will multiply the inverse of private key to cipher matrix according to the equation (4) and the result is obtained. By calculating the inverse of unimodular matrix and calculating it according to the equation (5), the message will then be decrypted.

Results

```
n ==[10,50,100,150]
serial=[0.0070742,0.0458006,0.11834478,0.245316]
openMpi=[1.96695,6.9930,52.64,362.299]
cuda=([1.289920,5.9012,15.5872,33.7572])
```



Limitations and possible improvements

For evaluating the encryption level of this image, we will focus on the differential attack. In this attack, attacker tries to understand the relationships between the plain and the cipher images. The attacker studies the effect of changing in one pixel of the input on the whole pixels of output cipher image to determine the key. To measure the effect of each pixel on the whole encrypted image, we can use three common security evaluations metrics:

1. Number of Pixels Change Rate (NPCR):

$$NPCR = \sum i,jD(i,j)W\ \ H\ 100\% \quad (8)$$

Where D (i.j) is the value of pixel difference in plain and cipher images at position (i,j) and W is the width of the matrix and H is the height of the matrix.

2. Unified Average Changing Intensity (UACI):

$$UACI = 1WH[\sum i,jc1(i,j) - c2(i,j)255]\ 100\% \quad (9)$$

Where C_1, C_2 are two ciphered images that their corresponding original images have only one pixel difference. Notice that C_1, C_2 are in the same size. $C_1(i, j)$, $C_2(i, j)$ are gray-scale values of the pixels at grid(i,j). $D(i, j)$ is determined by $C_1(i, j)$ and $C_2(i, j)$ if $C_1(i, j) \neq C_2(i, j)$ then $D(i, j)=1$; otherwise, $D(i, j)=0$(W and H are the columns and rows of the image).

3. Avalanche effect which shows the number of bits that are changed and is calculated according to equation (10).

Avalanche =Number of flipped bits in cipher pictureNumber bits in cipher picture×100% (10)

To evaluate the GGH encryption algorithm, three above mentioned evaluation metrics are considered on the images of three groups containing 100 images, Group 1 are images with size 50×50, group 2 are images with size 100×100, and group 3 are images with size 200×200. The comparison results between the padding based GGH and standard GGH algorithms are illustrated in Table 3.

Table 3

Evaluation of NPCR, UACI and Avalanche effect metrics for images of three sizes using GGH snail encryption and standard GGH algorithms.

Method ->	Improvement			Padding based GGH			Standard GGH		
Size	NPCR%	UACI%	Avalanche%	NPCR%	UACI%	Avalanche%	NPCR%	UACI%	Avalanche%
50×50	98	34.88	44.3	100	35.67	54.66	2	0.79	10.96
100×100	98.99	35.29	45.4	99.99	35.66	50.9	1	0.37	5.5
200×200	99.49	35.6	44.75	99.99	35.8	48	0.5	0.2	3.25

Based on the results obtained from Table 3. for the standard GGH algorithm, changing one pixel does not have a deep effect on the cipher image. Therefore, the method requires further improvement which can affect all of the pixels.

According to results in Table 3, the applied padding can affect all the pixels and by changing only one pixel, all of the pixels in the encrypted image will be changed. Also, the Avalanche effect can show the affected bits in the encryption process are performed correctly. The complexity of the applied padding is O(n2) and the complexity of GGH encryption is O(n2), so the whole complexity of the proposed method is O(2n2). By adding a simple padding, the GGH image encryption will be dramatically improved and the cipher becomes resistant to both cipher text and statistical attacks.

From the results, it can be deduced that the performance measurements based on the three measurement criteria including (i) Number of Pixels Change Rate (NPCR), (ii) Unified Average Changing Intensity (UACI), and (iii) Avalanche effect are calculated. The results on three different sizes of images showed that padding based GGH algorithm has improved UACI, NPCR, and Avalanche metrics by almost 100%, 35%, and 45%, respectively, in comparison to the standard GGH algorithm.

In conclusion, this shows that the new proposed method (i.e., padding based GGH algorithm) can be regarded as an improvement to the its fundamental version (i.e., standard GGH algorithm) which acts poorly on all of the UACI, NPCR, and Avalanche metrics.

Conclusion

GGH is a simple public key crypto system which is based on the closest vector problem (CVP). It encrypts data in the matrix form and makes it suitable for image encryption. It works in different dimensions and different square matrices. Experimental studies indicated that changing one pixel in GGH encryption does not have enough effect on the whole encryption output, so a new method is proposed which can add padding to the plain image before applying GGH encryption process. This padding applies the forward and backward snail tour XORing to the plain image in order to make it ready for further process using the GGH encryption algorithm. According to the final results which correspond to the proposed method, changing one pixel affects the whole plain image and also affects the final cipher image reasonably good which is encrypted by GGH algorithm. That is, by changing one pixel, 99.99% of pixels of the cipher image will be changed and also it has a good avalanche effect about 55% which has improved the standard version by more than 45% which shows an acceptable improvement in the whole image encryption to be prune to possible attacks. Moreover, the unified average changing has also dramatically increased and as a result, the GGH snail encryption algorithm is robust and efficient for encrypting medical images in medical image security realm. However, this is a successful study in improving the standard GGH algorithm evaluation metrics especially in terms of avalanche effect, the complexities of encryption and decryption process have been double while compared to the standard version while other parameter remained constant. On the other hand, it is worth considering that the increased complexities can be ignored in comparison to the achieved success (*i.e.*, reach avalanche effect of more than 50%).The ability to invert large matrices accurately and quickly determines the effectiveness of a wide range of computational algorithms and products. GPU computing is ideally suited for massively parallel tasks as the thread creation and memory transfer overheads are negligible. We have redesigned the Gauss Jordan algorithm for matrix inversion on GPU based CUDA platform,tested it on different types of matrices of various sizes, and have shown that the time complexity of matrix inversion scales as n if enough computational resources are available. Even a small matrix of size 10 10 did better using the parallel CUDA Gauss Jordan algorithm.

References

1. Öztürk I., Sogukpınar I. Analysis and comparison of image encryption algorithms. Transactions on engineering. Comput. Tech. 2004;3:1305–1313. [Google Scholar]

2. Potdar V., Chang E. Disguising text cryptography using image cryptography.; 4th International Network Conference INC; July 6-9, University of Plymouth, UK. 2004. [Google Scholar]

3. Mitra A., Subba Rao Y., Prasanna S. A new image encryption approach using combinational permutation techniques. Int. J. Comput. Sci. 2006;1(2):1306–4428. [Google Scholar]

4. Socek D., Li S., Magliveras S., Furht B. Enhanced 1-D chaotic key-based algorithm for image encryption. In: IEEE/CreateNet SecureComm; 2005, September 5-9, Athens, Greece 2005. pp. 406-408. [Google Scholar]

5. Shuangyuan Y., Zhengding L., Shuihua H. An asymmetric image encryption based on matrix transformation. Communications and Information Technology, In: IEEE International Symposium on, ISCIT 2004. vol. 1, 2004, pp. 66-69. [Google Scholar]

6. Mao Y., Chen G., Lian S. A novel fast image encryption scheme based on 3D chaotic baker maps. Int. J. Bifurcat. Chaos. 2004;14(10):3613–3624. doi: 10.1142/S021812740401151X. [CrossRef] [Google Scholar]

7. Alsultanny Y. Image encryption by cipher feedback mode. Int. J. Innov. Comput. Inf. Control. 2007;3:589–596. [Google Scholar]

8. Cagnoni S., Dobrzeniecki A., Poli R., Yanch J. Genetic algorithm-based interactive segmentation of 3D medical images. Image Vis. Comput. 1999;17:881–895. doi: 10.1016/S0262-8856(98)00166-8. [CrossRef] [Google Scholar]

9. Shamir A. How to share a secret. In: Communcations of ACM NY, USA. 1979:pp. 612–3. doi: 10.1145/359168.359176. [CrossRef] [Google Scholar]

10. Zhao R., Zhao J-J., Dai F., Zhao F-Q. A new secret image sharing scheme to identitiy chaters. Comput. Stand. Interfaces. 2009;31:252–257. doi: 10.1016/j.csi.2007.10.012. [CrossRef] [Google Scholar]

11. Panigrahy S., Acharya B., Jen D. Image encryption using self-invertible key matrix of hill cipher algorithm. In: 1st International Conference on Advances in Computing, 21-22 February, Chikhli, India, .2008; pp: 1-4. [Google Scholar]