IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 19, Issue 3, Ver. IV (May - June 2017), PP 01-05
www.iosrjournals.org

Image Encryption using RSAA

Vishal Snedan Robertson [1], Krishnakant Vilas Redkar [2]
[1] *Student (B.E. Computer), AITD, Goa, India.*
[2] *Student (MScIT), PRIDE, Periyar University, Salem, Tamil Nadu, India*

Abstract: *Information security has become very important nowadays in storage and transmission of data. Images are used everywhere around us. The security of certain images from unauthorised users is important. We present a variable key length cryptosystem based on Block Cellular Automata. It encrypts images by dividing it up into blocks 50 X 50pixels and encrypting each block at a time. It provides high security even with a key which is a single character. It is highly robust as it produces similar distorted cipher images for any type of input image. Cipher images aren't distinguishable to the naked eye. It can be used to store personal images securely on one's system and large scope for storing images securely on a server or cloud storage. Brute forcing this algorithm will take a lot of time due to the immense rule space of Block Cellular Automata and due to the variable sized key.*

Keywords: *Key, Encryption; Decryption; Cipher Image; Cellular Automata; Block Cellular Automata; Margolus neighbourhood; Rule Generator, Add Key, Hashing.*

I. Introduction

The internet and information technology are sprouting swiftly. As a result, people are widely using interactive media in communication such as images, audios and videos. Images play a significant role in communication of the military, NSA and personal affairs. Since, these images may carry highly confidential information; hence these images entail extreme protection. When people wish to transfer images over an insecure network, then it becomes crucial to provide an absolute protection. In brief, an image requires protection against various security attacks. The primary intention of keeping images protected is to maintain confidentiality, integrity and authenticity. Different techniques are available for making images secure and one technique is encryption. Generally, Encryption is a procedure that transforms an image into a cryptic image by using a key. A user can then retrieve the initial image by applying a decryption method on the cipher image which is the reverse execution of the encryption process.

II. Literature Survey

The idea of block cellular automata (BCA) consists in replacing blocks of cells with blocks of the same size. At a particular step blocks do not overlap with each other. The blocks choice shifts every step. This project considers a square 2-color lattice where 2x2 blocks are replaced with other 2x2 blocks, then the block choice shifts in the diagonal direction by 1 cell. The boundary conditions are periodic. There are 16 types of 2-color 2x2. So there are 16^{16} = 18446744073709551616 rules possible in the general rule space of the 2-color 2x2 block 2D BCA. Although all of them aren't reversible rules which can be used in cryptography.

Each iteration of the BCA operation consists 2 phases.

Phase 1: Sub-blocks of size 2x2 are created starting from cell at row 0 & column 0.

Decimal value of each block is obtained and is used as an index in the Rule to obtain the value to be mapped back to the block.

Phase 2: Sub-blocks of size 2x2 are created starting from cell at row 1 & column 1.

After obtaining the decimal equivalent of each block, it is used as an index in the Rule to get the value to be mapped back to the block

Forward Block Cellular Automata (FBCA) performs in each iteration Phase 1 followed by Phase 2.

Reverse Block Cellular Automata (RBCA) performs in each iteration Phase 2 followed by Phase 1.

Said Bouchkaren and Saiida Lazaar [2] explained an algorithm to use Block Cellular Automata to encrypt and decrypt text. Their algorithm makes use of a single rule which doesn't provide much robustness.

Marcin Seredynski and Pascal Bouvry in the Reversible Cellular Automata section, stated that a large number of reversible rules should be used in cryptography and that they should exhibit a complex behaviour. BCA has a rule space of 16! Rules i.e. 20922789888000 rules. From this rule space, there are numerous reversible rules that can be used to encrypt and decrypt data.

The authors of "Theory and Applications of Cellular Automata in Cryptography" [4] have proposed an algorithm which performs XOR on the message and the key and introduces a lot of randomness without increasing the

overall time.
The authors in "A novel cellular automata based technique for visual multimedia content encryption" [6] have designed an algorithm based on Cellular Automata whose output closely resembles that of the designed algorithm based on Block Cellular Automata

III. Proposed System

The designed cryptosystem is used to encrypt and decrypt images based on a single shared key. It is based on Block Cellular Automata. The algorithm generates 16 BCA rules that are based on the key entered by the user. There are a total of 16! = 20922789888000 reversible rules in BCA. The key used to encrypt and decrypt images can be of any length. The image is encrypted a single block at a time. Each block is of 50x50 pixels. If needed, the image to be encrypted is padded and then split min blocks. Each block undergoes 16 iterations wherein during each iteration a single rule generated during RuleGen () is applied on a single block of red, green, blue or alpha values of the pixels in the current block. The designed algorithm uses functions defined in RSAA Symmetric Key Cryptosystem [1]. Functions such as RuleGen(), HashKey() and AddKey() are used as they are defined by the authors. Changes to the encryption and decryption function are made so as to encrypt and decrypt images.

3.1 Rule Generator

The RuleGen() function is applied on the key after applying HashKey() function on the key. The secret key entered by the user can be a sequence of characters of any length. After the key being hashed, its length is now 1024bits. A total of 16 rules can be generated from the key as every 8 characters in the key generate a rule of Block Cellular Automata. The proposed Rule generation algorithm is as follows.
Algorithm RuleGen(Key)
numRules is the number of rules generated (every 8 characters of the Key creates a rule)
binKey stores the key in ASCII 8bit binary.
TempMat is a matrix of size numRules x 16
TempMat stores a 4bit hex representation of binKey.
RuleMat is a matrix of size numRules x 16.
RuleMat stores the final rules and is initialized to -1.
For every row of TempMat
index i is at start of the current row ie 0 and index j is at end of the current row ie 15
while i is less than j
Create a pair of the element at i and j which has not been paired before, if a pair can't be made then shift i to the right or j to the left or both.
Store element at j, at location i of current row of RuleMat
Store element at i, at location j of current row of RuleMat
For every location of current row of RuleMat whose value is -1, store value of location at location of current row of RuleMat
Return RuleMat

3.2 Hashing function

The algorithm applied a hash function on the input key. The algorithm works even with weak keys as the hashing function applied on the key transforms any key to a fixed length output. Any hash function can be used. Hence various implementations can be based on different hash functions. SHA-512 is applied on the input key in the following manner

Algorithm HashKey(key)
h1 = SHA-512 on key
h2 = SHA-512 on h1
Return h1 + h2

3.3 XOR function

The AddKey() function performs a bitwise XOR on the bits in the block and corresponding bits of the key. The AdddKey() function is as follows.

Algorithm AddKey(MessageMat,KeyMat)
For every bit of MessageMat
MessageMat = KeyMat XOR MessageMat

3.4 Encryption function

Images can be in jpeg or png. Different images encrypted using RSAA will have similar looking cipher images. The image to be encrypted is read from the input path specified. Padding is done to make the dimensions of the image a multiple of 50. Hence 50x50 blocks of pixels of the image will be encrypted and decrypted at a time. After padding is applied the 24bit ARGB values of the image are read into arrays to be used for manipulation. The R, G, B and A components of a block undergo FBCA followed by an AddKey() step. This happens for all 16 rules generated. AddKey() uses a part of the key in each iteration. Similarly, every block is encrypted and once all are done, the cipher image is written to the output path specified. The ImageEncryption() algorithm is given below.

Algorithm ImageEncryption(InPath, OutPath, Key)
Key = HashKey(Key)
RuleMat = RuleGen(Key)
RuleMat is a matrix of size 16 x 16
Pad the image at InPath using a specific color so that the image can be divided into blocks of 50 x 50 pixels
Read RGB values of the image at InPath into Rmat, Gmat and Bmat
For each 50 x 50 block of image
    For every rule in RuleMat
    Apply FBCA transformation on Rmat, Gmat and BMat using current Rule
    Apply AddKey() on Rmat, Gmat and BMat and KeyMat using one of the 4 parts of the Key
Write RGB values from Rmat, Gmat and Bmat into an image at OutPath

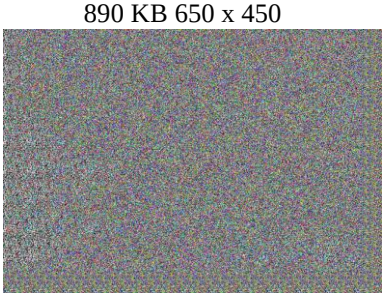
3.5 Decryption function

The image to be decrypted is read from the input path specified. The 24bit ARGB values of the image are read into arrays to be used for manipulation. The R, G, B and A components of a block undergo AddKey() followed by a RBCA step. This happens for all 16 rules generated in reverse order of encryption. AddKey() uses a part of the key in each iteration also in reverse order of encryption. Similarly, every block is decrypted and once all are done, the decrypted image is written to the output path specified. The ImageDecryption() algorithm is given below.

Algorithm ImageDecryption(InPath, OutPath, Key)
Key = HashKey(Key)
RuleMat = RuleGen(Key)
RuleMat is a matrix of size 16 x 16
Read RGB values of the image at InPath into Rmat, Gmat and Bmat
For each 50 x 50 block of image
    For every rule in RuleMat
        Apply AddKey() on Rmat, Gmat and BMat and KeyMat using one of the 4 parts of the Key in reverse order
        Apply RBCA transformation on Rmat, Gmat and BMat using current Rule
Remove padded pixels' values from Rmat, Gmat and Bmat
Write RGB values from Rmat, Gmat and Bmat into an image at OutPath

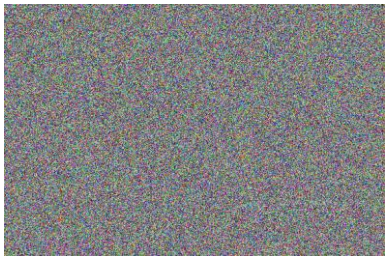
IV. Results

Here are sample images which were encrypted and decrypted using the designed algorithm. Each image has been encrypted and decrypted using the key RSAA

Original Image	Encrypted Image	Decrypted Image
421 KB 610 x 403	890 KB 650 x 450	461 KB 610 x 403
		

581 KB 594 x 400

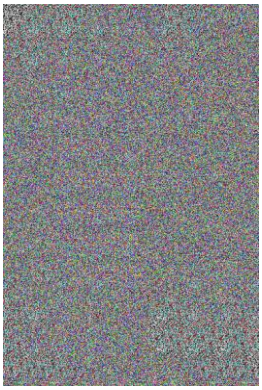
804 KB 600 x 400



472 KB 594 x 400

419 KB 400 x 594

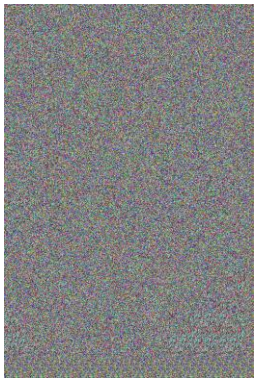
769 KB 400 x 600



389 KB 394 x 600

635 KB 487 x 703

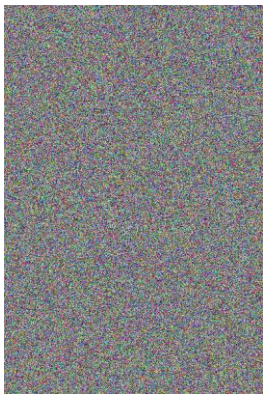
1.13 MB 500 x 750



618 KB 487 x 703

647 KB 394 x 600

805 KB 400 x 600



637 KB 394 x 600

The decrypted image dimensions match the original image. But as can be seen, the sizes of the original and decrypted image aren't the same. The difference isn't visually noticeable. Scenes of different types and even portraits all have the same distorted appearance after undergoing encryption. It is to some extent difficult to tell apart from two different cipher images.

V. Conclusion

The image encryption algorithm presented uses a variable size key and encrypts images using 50x50 block of pixels. Its security depends on the size of the key and the hashing algorithm used on the input key. Every type of image yields a similar distorted cipher image. Since this algorithm is not that quick, a brute force attack on it is computationally infeasible. This algorithm has immense application wherein image data security is of higher importance than time It exhibits the avalanche effect, in which a single character change in the input key leads to more than 80% change in output cipher image.

Applications
Store personal images securely on your system with a password.
Applications which store personal images of a user on a server, can encrypt and store them to safeguard from unauthorised access.

Future Scope
Work can be done to reduce the time taken by the algorithm by trying a faster implementation.
Other image formats can be made to work on RSAA.

References

[1]. Shreedatta Sawant, Vaishnavi Kamat, Vishal Snedan Robertson, Sneha Kamat, Anish Thali, Anuj Shetgaonkar, "RSAA Symmetric Key Cryptosystem" IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 19, Issue 3, Ver. III (May - June 2017), pg 53-57
[2]. Said Bouchkaren and Saiida Lazaar, "A Fast Cryptosystem Using Reversible Cellular Automata" in (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 5, No. 5, 2014
[3]. S. Nandi, B. K. Kar, and P. Pal Chaudhuri, "Theory and Applications of Cellular Automata in Cryptography" IEEE TRANSACTIONS ON COMPUTERS, VOL. 43, NO. 12, DECEMBER 1994.
[4]. Marcin Seredynski and Pascal Bouvry, "Block Encryption Using Reversible Cellular Automata"
[5]. Valloriw J. Peridier, "Basic Schemes for Reversible Two-Dimensional Cellular Automata", Complex Systems, 18, 2008, pg 47-48
[6]. Savvas A. Chatzichristofis, Dimitris A. Mitzias, Georgios Ch. Sirakoulis, Yiannis S. Boutalis, "A novel cellular automata based technique for visual multimedia content encryption", Complex Systems, 18, 2008, pg 3-7