**VISUAL CRYPTOGRAPHY USING CELLULAR AUTOMATA**

by

**Varshini.S(15BCE1130)**

**Shri Vignesh.S(15BCE1137)**

**Aashish.S(15BCE1140)**

**Moosa Aiman(15BCE1232)**

A project report submitted to

**Prof. Ganesan.R**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING(SCSE)**

in partial fulfilment of the requirements for the course of

**CSE4003 –CYBERSECURITY**

in

**B.Tech. (Computer Science)**

****

**VIT UNIVERSITY, CHENNAI**

**Vandalur – Kelambakkam Road**

**Chennai – 600127**

**NOVEMBER 2017**

**BONAFIDE CERTIFICATE**

Certified that this project report entitled “**Visual Cryptography using Cellular Automata”** is a bonafide work of **Aashish.S – 15BCE1140, Varshini.S- 15BCE1130, Shri Vignesh.S- 15BCE1137 and Moosa Aiman – 15BCE1232** who carried out the “J”-Project work under my supervision and guidance for the course **CyberSecurity**

**Prof.Ganesan.R**

**ACKNOWLEDGEMENT**

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof. Ganesan R,** Associate Professor, School of Computer Science, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

**MOOSA AIMAN VARSHINI S**

**SHRI VIGNESH S AASHISH S**

Visual Cryptography using Cellular Automata

**Aashish.S [15BCE1140]**

**Varshini.S [15BCE1130]**

**Shri Vignesh.S [15BCE1137]**

**Moosa Aiman [15BCE1232]**

**Abstract.** Visual cryptography depends on two shares. The initial configuration,extra security bits and the number of the rule for the CA along with the number of computed steps serve as a password for a visually encrypted image. The second share could contain a predefined pattern; the developed algorithm uses a snap-shot of a CA after a certain number of steps to generate the predefined share. Only one of these shares has to be random. The developed encryption system is a hybrid between visual and classical cryptographic approaches. It requires less storage space compared to a standalone visual encryption system and relies on Rule 30’s tested statistically significant randomness.

**Keywords:** *C*ellular Automata, CA Rule 30, Cryptography, One Time Pad,Visual Cryptography.

**1** **Introduction**

Visual cryptography is broad in definition and applicability, and there are many methods used in encrypting visual data. In essence, an image is converted into one or more images, which in isolation convey no information whatsoever. However, with a proper means to decrypt those images, one can display the original data.

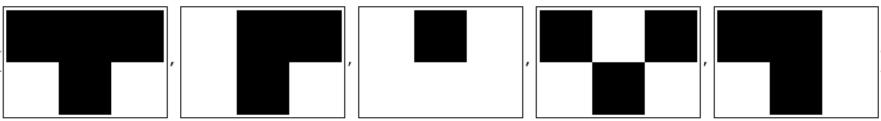
During Eurocrypt ’94, Moni Naor and Adi Shamir proposed a novel visual cryptographic method. Their method was based on the one time pad system of en-cryption. In its most basic form an image is split into two derived images or ‘shares.’ One share acts as a key and the other as a cipher. Each one, when viewed in isolation from the other, displays no meaningful data. However, when they are superimposed a discernible image can be viewed.

The advantage of their method lies in its security and practicality. It is completely secure due to the fact that without all the shares the original visual data cannot be retrieved. Also, the encrypted shares are generated in a random manner to ensure that no date can be retrieved from a single share. Its practicality on the other hand lies in computational decryption. Printing the shares on transparencies and superimposing them on top of one another will achieve the desired decryption.

Visual cryptography usually builds the first share in a completely random fashion. Then, using the original im age’s pixel data as well as the first share’s random pixel data, the second compleme ntary share is generated.

Despite the method’s si mplicity and practicality, the retrieval and decryption of the image requires presence of both shares. If one of the shares is missing, the decryp tion process becomes impossible. In addition, a single image of size n bytes has to be ex-panded, first by doubling each side and then by multiplying that by two.

If we can find a method that can deterministically generate a share based on a rela-tively small password, we can bypass the need for having two shares altogether. Our study proposes that Cellula r Automata (CA) rules can be used to grow a share based on a predetermined start state. A cellular automaton can be very easily represented in a visual way as a bi-dimensional square grid, or composed by lines of cells. Each cell is either black or white, and in every subsequent step (or line), there is an applied rule that dictates which color that specific cell will be, based on the previous state color and its immediate neighbors. See Fig. 1 as an example of a cellular automaton.



(a) (b) (c) (d) (e)

**Fig. 1.** Grid showing an example of a cellular automaton rule. The rule makes a particular cellwhite if either of its neighbors are white (c) on the step before, and makes the cell black if both its neighbors were black (a, d) or black and white (b, e).

Differentiating from other works in security using CA, our approach utilizes CA to address Visual Cryptography’s issues, with the aid of Rule 30, by only requi ring one share to be stored. By d oing so it helps reduce the overall size of the encrypted im age and preserve the ability to re store the original image with only one share.

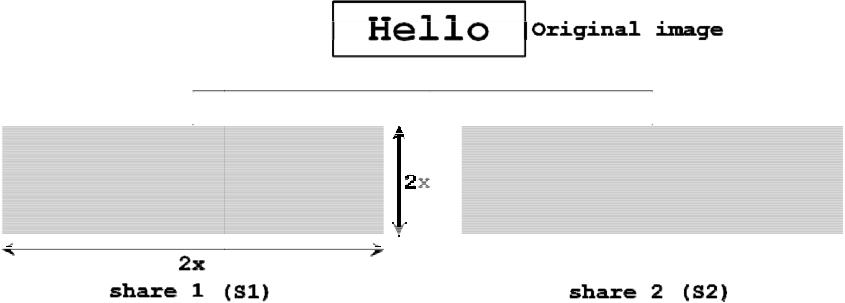
Despite the abundance of computational capabilities, timing brute force attacks could become an intensiv e computational task, increasing the time and processing exponentially, unlike context-free grammar passwords and the use of password dictionaries. Many times, the strength of written western alphanumeric passwords is related to personal information such as common names and stand ard file names in Unix. Additionally, employment of social engineering techniq ues can give attackers information about user passwords.

**2** **Approach**

**2.1 Original Visual Encryption Method**

The original encryption method is derived from the one time pad visual encryption system proposed by Naor a nd Shamir at Eurocrypt ’94. Its most basic form, a black and

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |



**Fig. 2.** This shows th e generation of two encrypted shares from a single image

white image, is split into t wo shares (S1 and S2). Both S1 and S2 will have twice the length and width of the orig inal image (see Fig.2).

Therefore, a single pixel in the original image is split into a set of four pixels. Any se t of four pixels derived from the original pixel will alternate in color: black-white-white-black or white-black-black-white (see Fig. 3-a).

The specific order of the pixels in each four-pixel set in S1 is generated rando mly. The order in the complementary share, ‘S2’, however, is not random. S2 is gener ated based on the original pixel i n the original image as well as the color and order of pi xels in S1 (see Fig. 3-b).

If the original pixel is b lack, the two complementary pixels, when superimpo sed, should produce four subse quent black pixels (see Fig. 3-c). If, on the other hand, the original pixel is white, the superimposed pixels should still alternate producing a pseudo-grey color resulting from the alternating black and white pixels.

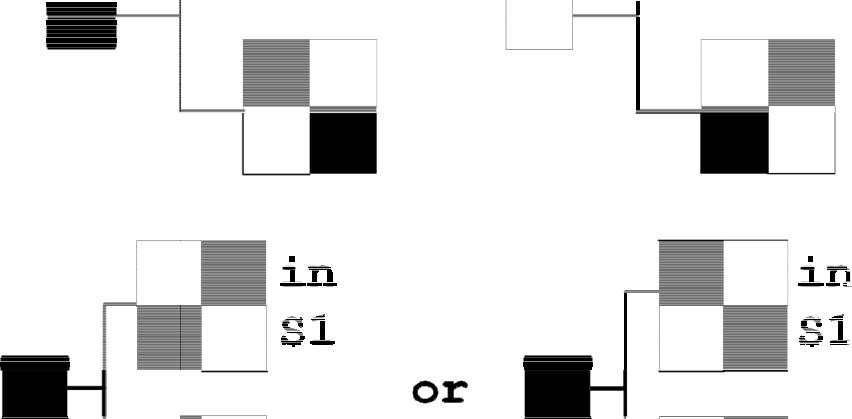
As a result, it is clear that this method is completely secure since the share is initially randomly generated. There is no way to tell whether a set of four pixels in an encryp ted share is derived from a black or white pixel by just looking at one share. Fig. 3 s um-marizes the visual encoding process.

**2.2** **Cellular Automata**

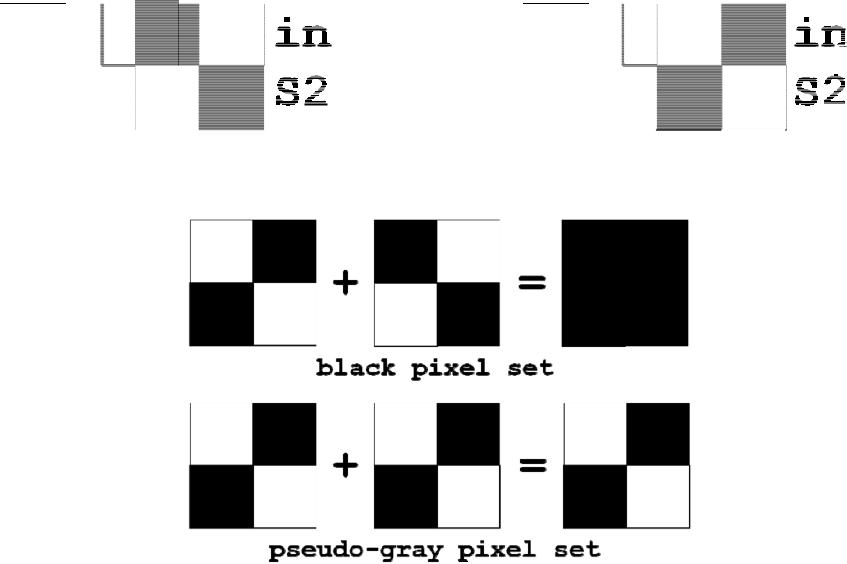
A Cellular Automaton is a discrete model that represents a number of parallel entities that interact with one anoth er in a way that influences their evolution and development. Simply put, a binary Cellul ar Automaton can be represented by an n-dimensional a rray where each member (i.e. cell) is an entity. Each entity normally has one of two state s, 1 or 0. The state that a cell assumes in a given iteration is dependent on a set of rules that govern how a cell evolves from one iteration to the next in accordance with the ce ll’s state and its neighbors’ states.



(a)



(b)

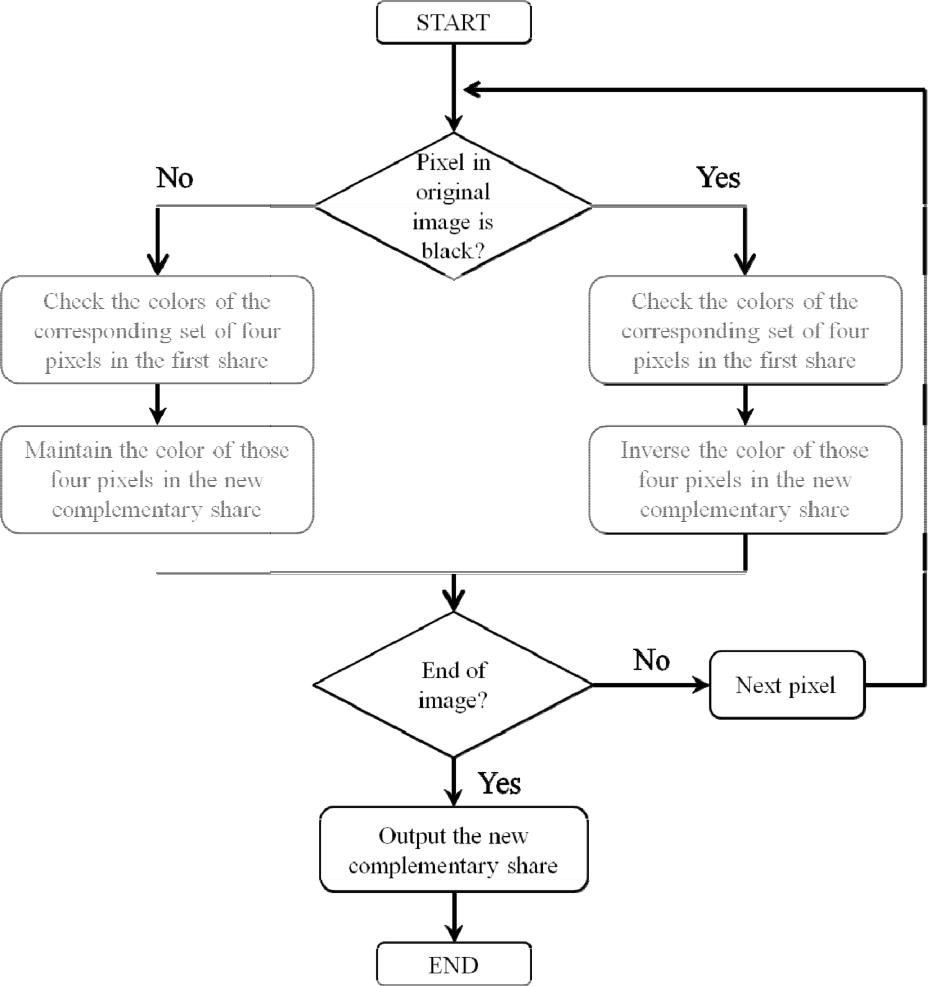


(c)

**Fig. 3.** (a) Possible sets of 4 pixels generated in share 1 and share 2 from encrypting a white orblack pixel. (b) Two possible sets of 4 pixels generated in share 1 and share 2 from encrypting a black pixel. (c) The result of s uperimposing corresponding sets of 4 pixels.

This field of computational sciences dates back to the mid 1900s where it was the interests of the likes of Sta nislaw Ulam, John von Neumann and John Conway. It is very impressive in its capa city to generate complex patterns from a simple set of r ules and conditions. This capacity is utilized here to grow a seemingly random share by specifying a password as t he initial condition and using a CA rule to grow the share. The specific rule chosen to carry out this task is rule 30.

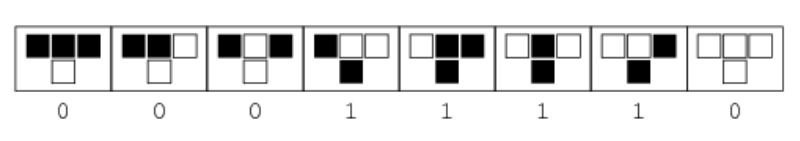
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |



**Fig. 4.** Sum mary of the proposed visual encoding process

**2.3** **Rule 30**

Rule 30 is one of the one-dimensional binary CA rules introduced by Stephen Wolfram in 1983. A cell’s sub sequent color is specified by the color of its immediate neighbors. The rule is nam ed “Rule 30” (see Fig. 5) because the rule outcomes are encoded in the binary representation of 30 = 000111102.



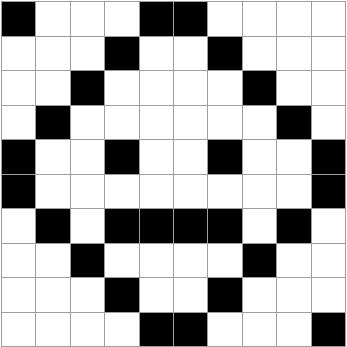
**Fig. 5.** Rule 30. This rule is used to grow a share that acts as the key to both encryption anddecryption of an image.

**2.4** **Visual Encryption via Cellular Automata**

In the original visual cryptography methodology, the first share is produced randomly, without any predefined rule or pattern. Following the typical visual cryptography method, the second and complementary share is generated based on the original im-age’s pixel information and the pixel pattern in the first share. As a result, every visual encryption needs to have both shares in order to generate the original image. Con-trasting with the common visual cryptography, our proposed method uses the CA rule 30 and only one share from the original image, instead of the common method in visual encryption where two shares are needed.

One of the many capabilities of a CA is that of generation or growth of patterns based on a simple set of initial rules. Dr. Stephen Wolfram provides evidence that randomness is found in the sequence generation in the time sequences that are created from running certain CA rules. This capacity of random production can be ex-ploited in visual cryptography so we could generate a complete share, starting with no more than a relatively small password taken from an image.

To encode an image using this method, a password would need to be generated. As an example for proving this method, the password is predefined to be 100 bits as a minimum size. The value of the password is extracted from the first row of pixels in the original image. If the first row is not sufficient to generate a 100-bit password, random bits are padded on to complete the 100 bits (but as a practical matter, this is not restricted to that number of bits, and could be greater than that). The CA rule 30 initial conditions are (i) the randomized first row and (ii) the initial two-color pixelized pic-ture (see Fig. 6), with the picture’s initial width and height.



**Fig. 6.** Original Image: Picture= {{1,0,0,0,1,1,0,0,0,0}, {0,0,0,1,0,0,1,0,0,0}, {0,0,1,0,0,0,0,1,0,0},{0,1,0,0,0,0,0,0,1,0},{1,0,0,1,0,0,1,0,0,1},{1,0,0,0,0,0,0,0,0,1},{0,1,0,1,1,1,1,0,1,0},{0,0,1,0 ,0,0,0,1,0,0},{0,0,0,1,0,0,1,0,0,0},{0,0,0,0,1,1,0,0,0,1}}

ECA rule 30, with a window size of 200, appears to be adequate as a random number generator, passing all but one statistical test out of complete set of NIST statistical tests for randomness.

In the implementation of this methodology, the encryption and decryption process were programmed in C/C++. The following pseudocode is based on the C/C++ that demonstrates how that was achieved starting with the original image as shown below.

Splitting the image into key and cipher

int i,j;

for(i=0;i<n;i++)

{

for(j=0;j<n;j++)

cout<<arr[i][j]<<' ';

cout<<endl;

}

}

void split(int arr[][5],int n)

{

int i,j,a,b;

for(i=0;i<n;i++)

{

a = 2\*i;

for(j=0;j<n;j++)

{

b = 2\*j;

if(arr[i][j] == 1)

{

cipher[a][b] = 0;

cipher[a+1][b] = 1;

cipher[a][b+1] = 1;

cipher[a+1][b+1] = 0;

}

else

{

cipher[a][b] = 1;

cipher[a+1][b] = 0;

cipher[a][b+1] = 0;

cipher[a+1][b+1] = 1;

}

}

}

for(i=0;i<n;i++)

{

a = 2\*i;

for(j=0;j<n;j++)

{

b = 2\*j;

key[a][b] = 1;

key[a+1][b] = 0;

key[a][b+1] = 0;

key[a+1][b+1] = 1;

}

}

}

int main()

{

int i,j;

int n = 5;

int a[5][5],b[5][5];

fstream infile,cp,k;

infile.open("C:\\Users\\USER\\Desktop\\input.dat");

cp.open("C:\\Users\\USER\\Desktop\\cipher.dat");

k.open("C:\\Users\\USER\\Desktop\\Key.dat");

for(i=0;i<n;i++)

for(j=0;j<n;j++)

infile>>a[i][j];

print(a,5);

split(a,n);

for(i=0;i<2\*n;i++)

{

for(j=0;j<2\*n;j++)

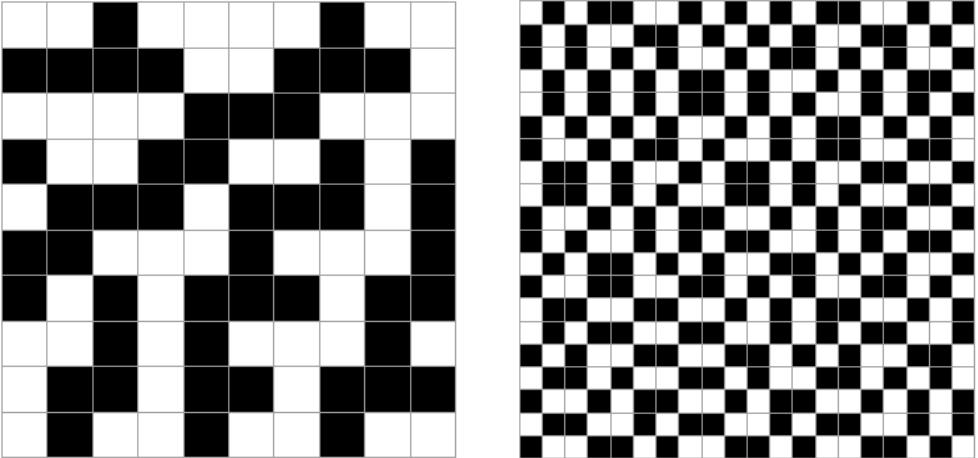
k<<key[i][j]<<' ';

k<<endl;

}

}

Afterwards, the second share can be generated using the original image’s pixel data and the relative organization of pixels in the CA grown share, which is similar to the original Visual Cryptography method, being able to utilize one single share (Fig. 7).

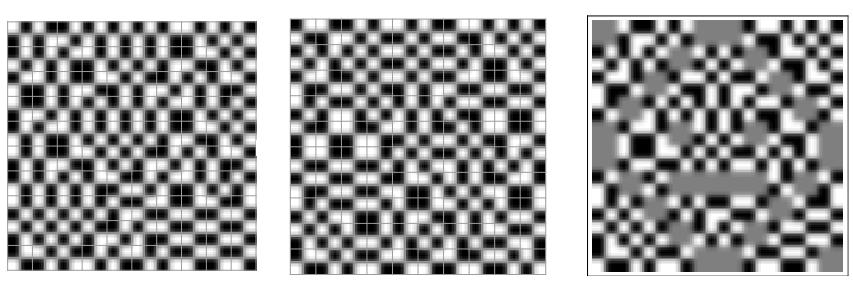


(a) (b)

**Fig. 7.** (a) Generation of a random share from the CA rule 30 and initial conditions. The mini-mum password size = 100 bits regardless of original image size. (b) The random share with each pixel represented by 4 alternating black and white pixels.

**2.5 Visual Decryption via Cellular Automata**

The output of the encryption process is a single share (Fig. 8). The password used for the encryption process is needed to decrypt the share and retrieve the original image. That password along with rule 30 is used to generate the other share needed to produce the original image. Similar to the initial stage in the encryption process, the password is used to grow the share. That share’s pixels are then represented by four pixels each. Finally, the original image can be retrieved by superimposing the CA grown share with the single encrypted share. In a post-processing method, the image can be cleaned up by just changing the color of the gray into white.



**+** **=**

(a) (b) (c)

**Fig. 8.** Visual Decoding Process: the addition of the shares (a) and (b) reconstruction of theoriginal image (c)

**Discussion**

The original Visual Cryptography method usually relies on a one-time pad encryption method. Two shares are generated, where one acts as a cipher while the other acts as a key. Both the cipher and its respective key have to be present in image formats. Moreover, each of those images is double the width and double the height of the original image’s dimensions. The resultant size increase per image ends up at eight times the original image’s size. This is a large size increase in addition to the incon-venience of storing two separate shares per image.

Our method solves these two issues. We only need to store a single encrypted share and that share acts as our cipher. To decrypt the share, a visual key needs to be present. Unlike the original method, however, the key does not have to be stored as a full share. We only need the original 100-bit password, and from it an entire visual key can be grown via CA Rule 30.

This method is not more convenient as a result of storing a single share rather than two, but also more efficient. Only a single share needs to be stored, which is four times the size of the original image and half the size required by the common method.

**CONCLUSION**

The method described here builds on the original Visual Cryptography method pre-sented by Naor and Shamir. It utilizes CA’s ability to generate pseudorandom patterns from a simple starting point to grow a share from a predefined password. This approach is both more convenient and efficient than the original Visual Cryptography approach.