Fast Image Encryption with a Cellular Automata

Shin Lee
Western Reserve Academy

Abstract:
Due to the improvements of cryptology based off of the RSA encryption algorithm, modern society can communicate with each other safely online. However, when transmitting heavy data such as photos or videos, the RSA algorithm will be overwhelmed with the overhead of computing huge prime computations. In this paper, I have developed a methodology to transmit images securely like the RSA,but with faster processing time. To achieve this, I have utilized 2D Cellular Automata and its characteristic that when invertible rules are applied they can be used to encrypt and decrypt data. An experiment of this algorithm conducted at the end resulted in more than ten times faster processing time for encryption/decryption than the RSA.

1. Introduction

Motivation

In line with my interest in algorithms, I was watching a Youtube video on algorithms that changed the world, in which the RSA algorithm was introduced. I was surprised how such a simple mathematical concept could be applied to enable people around the globe to safely communicate with each other.

Intrigued by the concept, I was motivated to come up with a new method that also employs this public key encryption system. The RSA encryption algorithm uses huge prime numbers to generate a public key and a private key, which is complicated and relatively slow. This makes RSA unfit for processes with large amounts of data, such as transmitting pictures or videos.

In line with my interest in algorithms, I was watching a youtube video about algorithms that changed the world, where I found out about the RSA algorithm. How the RSA algorithm enabled a very secure interaction with only a simple mathematical base was truly fascinating. It inspired me to come up with a new encrypting system with the same public-private key system used in the RSA algorithm. I eventually came up with cellular automata, a concept I learned during an art project before, to try to come up with an algorithm to encode an image. Interestingly, cellular automata can encode and decode very fast when given a rule to start with, and is especially useful in encrypting data that has pixels such as image files. In this paper, I created an image encryption algorithm that utilizes cellular automata to fastly and robustly encrypt data.

Design of the Paper

This paper will discuss various encrypting logics and the RSA which utilizes a public key system. Then, it will contemplate on basic Cellular Automata and their characteristics. Using this knowledge, in Part 3 I will prove several characteristics that are essential in encrypting images, and come up with a custom algorithm and demonstrate the encrypting and decrypting operations on images.

Background Knowledge

RSA Algorithm [1]

The RSA Algorithm is one of the security measures taken in order to prevent unintended decryption from happening. It utilizes the fact that operations to find a prime number is one of the most time-consuming ones, and also using the fact that prime numbers have a unique characteristic that will be described below that can be used to make a secure method to transfer information.

The RSA Algorithm works as such:

1. Unique prime numbers p and q are chosen, and $N=pq$ is made public
2. Assuming that there is no common factor between $(p-1)$ and $(q-1)$, a number e is chosen in the range from 1 to min(p-1, q-1). The number e is made public.
3. A private key d is derived from the formula $ed = 1 \bmod (p-1)(q-1)$.

After these steps, a public key pair (N,e) and private key d is used to encrypt/decrypt data.

When sending a message, we utilize the public key to encrypt it. Suppose that we want to send a message that has been numericized as M. Then, the encrypted result will be $M^e \pmod n$

When the receiver receives the message, then the person can decrypt using the private key, that

$$M^{ed} = m \,(mod\ pq)$$

The RSA's method of encryption is a good example of asymmetric encryption, in which a public key, used for encrypting, and private key, used for decrypting, are different. This asymmetric encryption is very safe in terms of communication, as anyone can encrypt a message but only the receiver with the private key can decrypt them. [2]

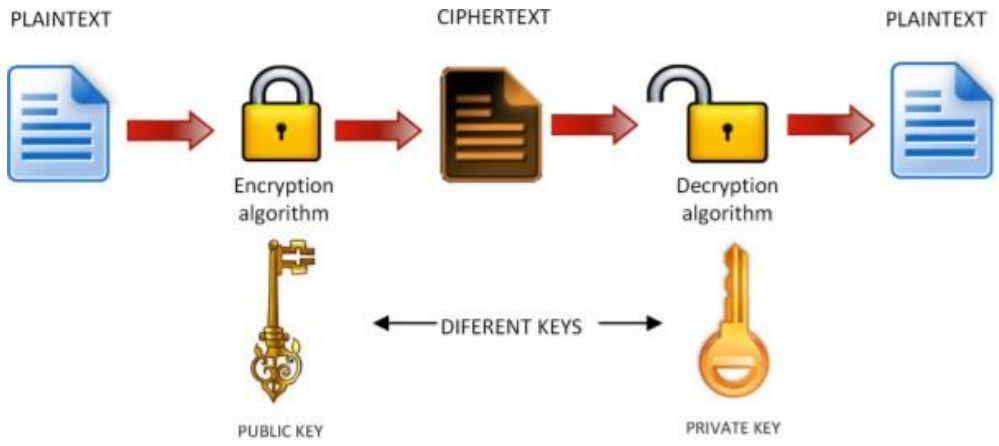


Figure 1: RSA Encryption Algorithm's utilization of public key and private key

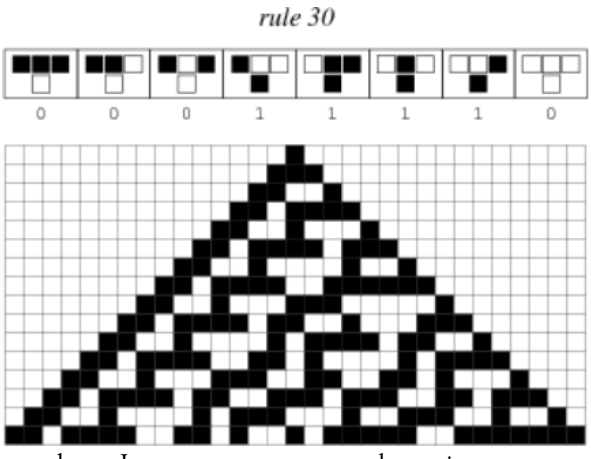
Cellular automata [3]

Cellular Automata is a sort of grid that consists of numerous cells. Each cell has a value of either 0 or 1 (white or black), and when given an initial state, their next state is determined through a set rule. Each rule of cellular automation can be represented in a decimal number, where we convert it to binary to determine the next status of each instance based on its neighboring instances.

The value of the cell (i,j) is determined by the values of the cells (i-1, j-1), (i-1, j), (i-1, j+1) For example, the figure below is depicting the logic of rule 30. If rule 30 is converted into a binary number, it becomes 00011110. Each digit in that number represents the outcome when the previous cells are each 000, 001, 010, 001, 100, 101, 110, 111. When this is repeatedly applied to the first row, we get this beautiful pattern as depicted below.

Figure 2: Depiction of rule 30 on Cellular Automata and its change based on initial state

Due to its unpredictable nature, it is often used in cryptography and generating random



numbers. In our paper, we are also going to use this feature to encrypt a selected image data's individual pixels.In this process, we utilized the 2D Cellular Automata, where if we apply mutually invertible rules, we can encrypt and decrypt data. In addition, to prevent attacks, we have come up with a random sequence to apply CA multiple times instead of a singular encryption to increase security.

3. Experiments

3.1 Finding Invertible CAs

Regardless of which encoding method, there must be a way to decode the encrypted data. Thus, a function that converts a message into an encrypted one must be invertible. Therefore, we must first find out which CAs are invertible. Thankfully, there are only 256 rules that are present in CAs, so we can check if each rule is invertible through a brute force algorithm. The code to do so is the one below.

```
ruleNumber = 256

for rule1 := 1 ~ 256:
  for rule2 := 1 ~ 256:
    test := {000, 001, 010, 011, 100, 101, 110, 111}

    if applying rule2 after rule1 to test is same with
    test:
      {rule1, rule2} is an invertible pair
```

Figure 3: Bruteforce algorithm to find invertible rules

Through this experiment, I found six invertible rule sets. They were {15,85}, {51,51}, {85, 15}, {170,240}, {204,204}, and {240, 170}. When applying this to use in production, we might want to go further than just 256 rules to keep brute force attacks by hackers.

3.2 Encryption

After finding invertible CAs, we have to make a complicated encryption method based on that. If we were to just simply apply a few CA rules to encrypt, hackers would easily grasp our original image by brute forcing through the possible rule sets. To prevent this, I came up with an encryption method involving multiple layers. First, I divided the images used in the experiment of size 64x64 pixels to 8x8 sized blocks of images. Then, on those individual blocks, I considered them a 64-bit string instead of a matrix and applied CA. However, we are going to use different rules to do such in each of the blocks. During this process, we are going to use a randomly picked rule pair that is invertible to block brute force attacks.

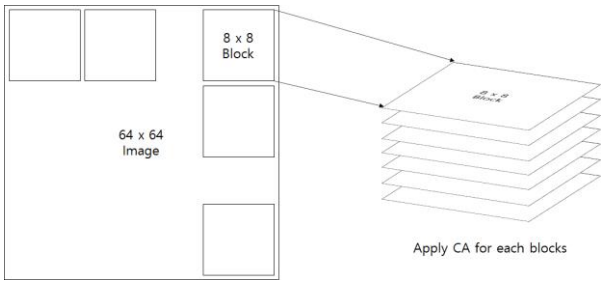


Figure 4: Dividing up 64x64 sized image into multiple blocks to encrypt

3.3 Decryption

In order to decrypt a given image, we need to find out which rules were applied in the encryption process. For that, we need to know how many rules are there, and which of those invertible rules were applied on each blocks. In this experiment, we restricted the rule size to 256 and chose one of the 6 invertible ruleselts, but when in real implications, we will need to choose larger limits, such as 1024 rules,for example. When image is sent, the sender will transmit a sequence of numbers that contain informationof which rules were used. This information will be sent using the RSA algorithm, so it is safe from attack. After all sequences has been sent, we send the encrypted image to the receiver. We are not going to use the RSA algorithm here, because it takes too long to encrypt larger data like images using RSA. The receiver can decrypt the image by using the sequence received beforehand reversely. Thus, we used RSA's security but resolved its slow speed by applying cellular automata in the process. Even if the hacker intercepts the encrypted message, there are too many cases to brute force to guess the sequence (if there are 10 blocks in 256 rule sets, there are 256^10 cases this is possible), thus making it safe from attack.

Here are the actual images encrypted and decrypted in the experiment.

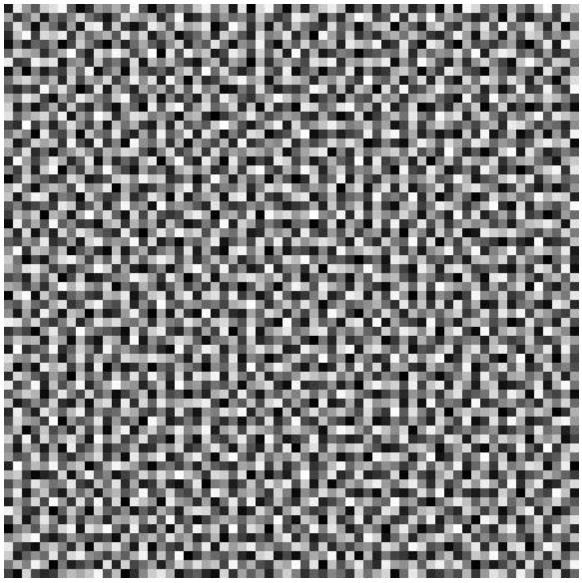


Figure 5: Encrypted image data using CA and RSA [4]

3.4 Performance

In order to compare the performance, we have measured the speed of using just RSA versus my method of utilizing cellular automata on 20 images. Each image was 64x64 pixel sized, and is a black and white image that has value between 0~255. All models were created using Python, and the experiment was conducted using a CentOS Linux machine with an Intel Xeon E5-2680 V3 (@ 2.50GHz) Dual-core CPU, paired with 256GB of memory. The result of the experiment, as shown below, shows that using cellular automata in the process makes encryption about ten times faster than using just RSA.

	RSA	CA
Encryption	253ms	27ms
Decryption	178ms	21ms

Figure 6: Comparison between average processing time of RSA and CA method

4. Conclusion

In this paper, we have come up with a method to safely transmit data like the RSA algorithm but with much faster speed. In this process, we utilized the 2D Cellular Automata, where if we apply mutually invertible rules, we can encrypt and decrypt data. In addition, to prevent attacks, we have come up with a random sequence to apply CA multiple times instead of a singular encryption to increase security.

As a result, it has turned out that the method utilizing CA is ten times faster than just using RSA in encrypting and decrypting images. This is because while RSA has to go through heavy computation using prime numbers, our method applies Cellular Automata's rules to encode, which is relatively simpler. In the future, it would be a good idea to research about encrypting text data or bit sequences such as video using the same concept.

References

[1] Josh Lake, Comparitech, "What is RSA encryption and how does it work?" URL: https://www.comparitech.com/blog/information-security/rsa-encryption/

[2] Margaret Rouse, TechTarger, "RSA algorithm (Rivest-Shamir-Adleman)"
URL:
https://searchsecurity.techtarget.com/definition/RSA

[3] Wolfram MathWorld, "Cellular Automaton"
URL:
https://mathworld.wolfram.com/CellularAutomaton.html

[4] Image of Lena Forsén
URL:
https://sipi.usc.edu/database/database.php?volume=misc&image=12