MANIPAL INSTITUTE OF TECHNOLOGY

(A constituent Institute of MANIPAL UNIVERSITY)

MANIPAL - 576 104, KARNATAKA, INDIA

Project Report

On

A TRNG-Based Recursively Seeded Image Encryption Algorithm.

Submitted By

1. **Soorya Annadurai - 140905116**
2. **Nishith Sinha - 110905678**

August 2014 – November 2014

Abstract:

Encryption, in its simplest form, is a process of masking valuable information, data, or content, so that only a select few can see the true content. It has been practiced for ages, but it’s most popular application has been in the military. Secrecy has been a pre-requisite for all military communications, and newer and newer methods of encryption have risen over the years.

The primary reason for the development of newer and newer algorithms is to increase the difficulty level for an unauthorized person/entity to unmask the secret information by any means other than the intended decryption process. That is to say, it is an attempt to remove any other method of “cracking the code”.

In the past, the best algorithms have been brought down by the recognition of a pattern in the algorithm. By analysing the encrypted/decrypted data, it is the deduced pattern that usually leads to the unravelling of an entire algorithm.

For example, in World War II, the Germans under Adolf Hitler developed an encryption machine called Enigma. Due to the Enigma's statistical security, Nazi Germany became overconfident about their ability to encrypt secret messages. This overconfidence caused the downfall of the Enigma. Along with Enigma had several built in weaknesses that Allied cryptographers exploited. The major weakness was that its substitution algorithm did not cryptographers to decrypt a vast number of ciphered messages sent by Nazi Germans.

So, even the slightest pattern that can be found, can render an encryption algorithm useless. However, the intention of this algorithm is to remove any such pattern from an unauthorized party. In the course of this documentation, the actual process of how-to-do-so has been elaborated.

Introduction:

The main intent of this algorithm is to remove any observable pattern. By “observable”, I refer to a deduction from an external analysis, not merely regenerating the pattern based on a predefined start point.

The strength of an algorithm lies not in the complexity of the operations used on the original data, or the number of processes used to mask it. More often than not, it is the ability to hide the pattern used to generate the encrypted file. For instance, if the sequence “a a a a a” (5 a’s) were to be encrypted by using the most complex process, like multiplying them by a set of 50 different numbers that were virtually undetectable, and exchanging (permuting) the letters with each other, the result will still resemble “x x x x x”. Here, the process of regenerating the value of ‘x’ might be infinitely difficult, but a very obvious pattern can be recognized when the entire batch of “a a a a a” was encrypted.

However, if a very simple, one-step process of encrypting the data is used, but it is ensured that no pattern is observed by an unauthorized party, then the attempts to crack the encryption become that much harder, because they have no point to start from.

And, in terms of image analysis, the pattern does not restrict itself to the binary level. Even from a larger birds-eye view, histogram analysis, shape detection, and so many other forms of visual patterns can arise.

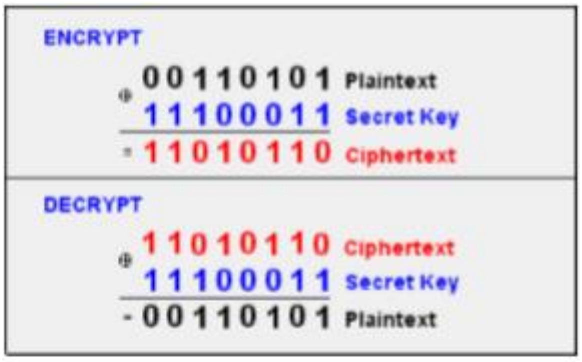
This algorithm addresses these issues in particular. It attempts to combine both the multi-layer encryption complexity, and the removal of any traces, both visual and binary. The approach is straightforward, but the security is uncompromised.

In this algorithm, I am focusing on the encryption of images in particular.

Every image is made of definite pixels. And each pixel is made of three colors, which are red, green, and blue. Different combinations of varying intensity of each primary color give rise to different colors. This algorithm is a way to manipulate the RGB values of each and every pixel in an image so that the final result shows minimal correlation with the original.

Methodology:

To substitute each pixel with another, I have chosen to use a simple xor-schematic. Using a particular key, the key is converted to its binary form, and then xor-ed with each value of R, G, and B of each pixel in the entire image. This is the substitution process.



Using the property that repeating the xor operation with the same key against the encrypted cipher can recreate the original pixel. The biggest advantages of this are that the xor operation can be used upon itself to regenerate the original data, and the size of the cipher does not magnify after the operation (in contrast with multiplying and dividing, where the size of the number undergoing the operation will vary).

The highest value of R, G, or B in a pixel is 255, and the lowest is 0. This means, it can be represented as an 8-bit binary value. To undergo the xor operation, it requires an 8-bit key. This key must be stored/remembered by the user.

The actual methodology, and how to close all the loopholes (pertaining to pattern recognition) are elaborated here.

This is the first step in encrypting the image. It helps in ensuring a one-many mapping system between the original pixel and the final encrypted pixel.

If we have the same key used in the xor operation for all the pixels, then every identical pixel will have the same output, no matter how random the key is. This is analogous to the example of “a a a a a” being mapped to “x x x x x”. So, the idea being used here is to create a list of random numbers, and use them as individual keys with each and every pixel.

For example, if we have a list of random numbers S = {N1, N2, N3, … , Nn}, and a corresponding set of all pixels in the image P = {P1, P2, P3, … , Pn}, we can use each element from S, and xor it with the corresponding element in P, to generate the encrypted set of pixels E = {E1, E2, E3, … , En}, which have the exact same (x,y) positions of every original pixel in S.

That is, Ni xor Pi = Ei,

And every element (pixel) of E goes to the same position in the final image, as the original.

But the difficulty here is to remember the entire list S. For a normal image, there can be tens of thousands of pixels, each having their own R, G, and B values. It is not feasible to enter the list S again and again to encrypt an image, although if it could be done, it would prove to be a very secure mode of encryption.

So, to simplify this task, a random number list can be generated at the time of encryption. To make sure this list is truly unique, but at the same time, reproducible (required at the time of decryption), the random number generation can depend on one random key, α. That is, α is the seed of the random number list.

The process of generating this random list is as follows:

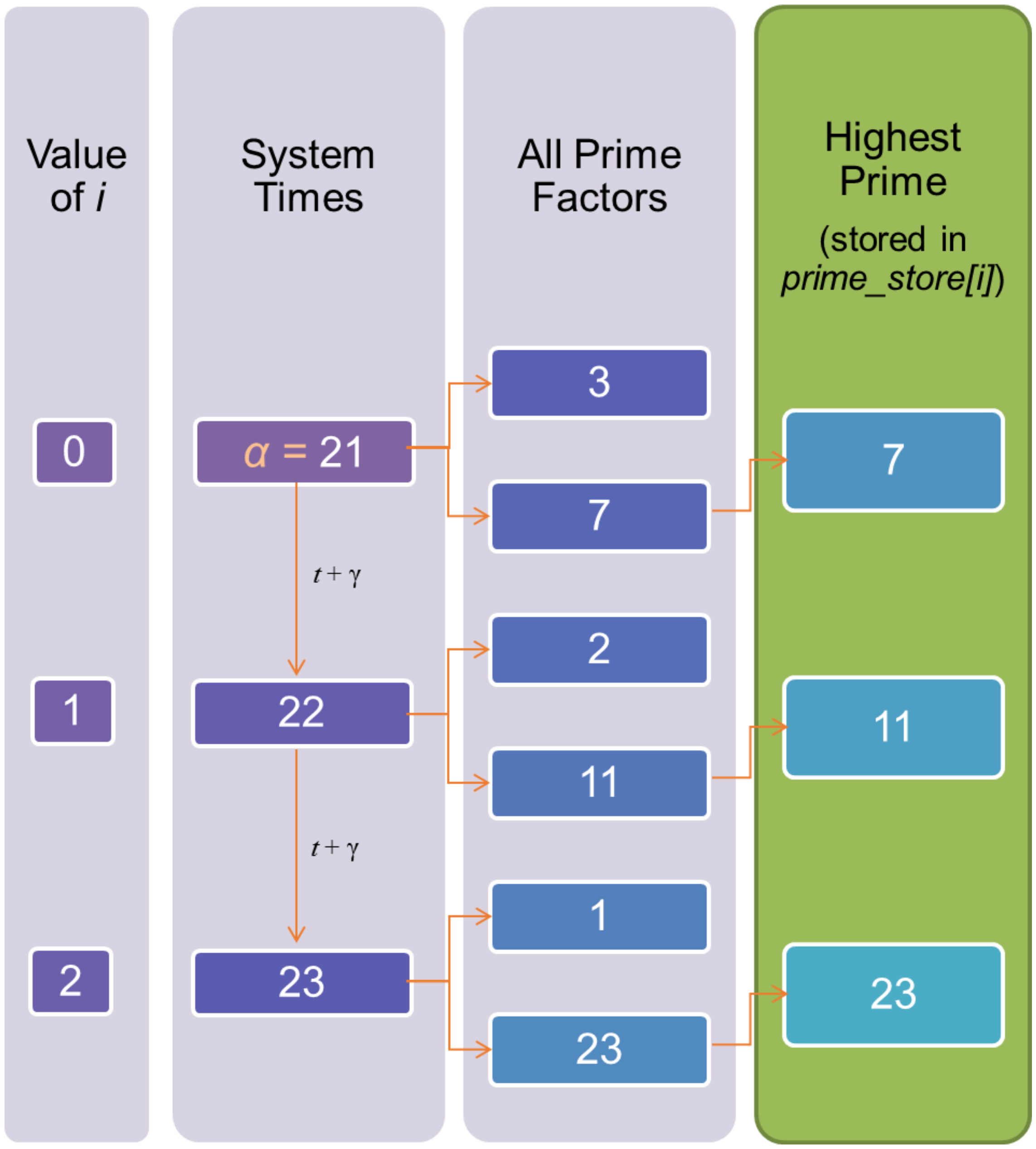
We take the value of α, and list out its prime factors. The highest prime value is extracted from this list, and stored as the first element in a prime store, S. This S is analogous to the set S taken in the previous example, which stores a set of random numbers.

After this is done, we create an increment factor γ (which could be generated from a user-defined password, for example), and increment the value of α by γ, ie. α = α + γ.

Then the factorization process continues, and another prime is generated. These subsequent primes are kept in a prime store. This process is iterated k times, which implies that we have k prime elements in the prime store.

Ideally, the value of k should be equal to n*3 (the number of pixels in the image, times the 3 color intensity levels). This means that every pixel's R,G,B values will be xor-ed with a different number, thus showing no traces of a pattern. But this isn't done, because factorizing that many large numbers {α1, α2, α3, …, αn} will take a significant amount of processing time. So we limit the size of the prime store to a much smaller value, k. The exact mapping of each element of the prime store with each element in the pixel set will be elaborated at a later stage.

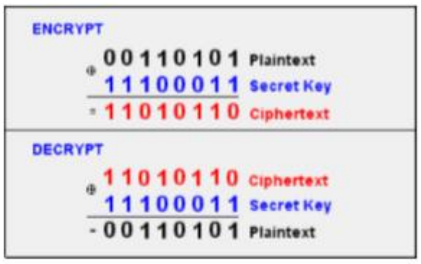
The seed α must be unpredictable (or random) in nature. This is to ensure the unique-ness of the subsequent prime store, generated from it. For example, α could be a password entered by the user, or the system time at the instant of encryption.



This flowchart shows the logical flow in generating the prime store of random numbers, S = {N1, N2, N3, … , Nk}.

Now, from S, we take the modulus of each element Si with 255, and then convert it to its binary form, to obtain random 8-bit sequences, that have no correlation with each consecutive byte. ie. create a new set B = {B1, B2, B3, … , Bk} where Bi = β(Ni % 255), where β is an operator to convert the argument to its binary form.

In this manner, we can obtain a list of random 8-bit values, which is very close to the initial target of



where we require the 8-bit random secret key to xor with each R,G,B value of every pixel.

So, at this stage, we have:

- The total number of pixels,
  n
- The size of the prime array,
  k
- An array of pixels,
  P = {P1, P2, P3, … , Pn}
  Where each pixel, Pi = {Ri, Gi, Bi}
- A random list of 8-bit sequences,
  B = {B1, B2, B3, … , Bk}

Now, we begin the xor-operations.

If Δ denotes the xor operation,

E(i)(j) = P(i)(j) Δ B((3*i + j) % k)

Essentially, each R,G,B value of the original image is xor-ed with every consecutive element of the random 8-bit list. And after the entire list (of k elements) is iterated through, it starts at the beginning of the list again.

In this manner, the first k elements are well encrypted. But if the (k+1)th element of the original image is identical to that of the 1st element, then the cipher will also be identical, as the same B(1) is xor-ed with the 1st and the (k+1)th element.

In simpler words, if the entire image consists of identical pixels, then every batch of k pixels in the encrypted image will be identical as well.

To overcome this issue, the list of random 8-bit sequences is modified every time it is accessed. That is, after every pass, the (i)th element will be accessed again and again, but the value of the (i)th element will always change.

In specific detail, for every element, after using it in one xor operation, it will be replaced by the square of that element, multiplied by the next element in the list.

Ie. E(i)(j) = P(i)(j) Δ B((3*i + j) % k) //as before
B((3*i + j) % k) = B((3*i + j) % k) * B((3*i + j) % k) * B((3*i + j <u>+ 1</u>) % k) // to make sure the list is never repeated.

So, every iteration acts as the seed for the next iteration, which is the concept being applied in the proposed algorithm.

Essentially, this simulates a list of random 8-bit binaries of size n, not k.

In this way, every R,G,B value of every pixel of the entire image is xor-ed with a different value.

And to provide an additional layer of protection for the encrypted image, the encrypted image E can be re-encrypted by iterating the pixel list from the reverse direction, to form the final double-encrypted image R.

That is, if μ is the encryption operation, as entirely discussed above, R(i) = μ(E(n – i))

Where R(i) is the (i)th pixel.

And the initial seed, α is provided to the user as a passkey to decrypt the same image.

The primary purpose of this operation is to provide an extra layer of protection for the first k elements in E. Since they were generated first, from the primes seeded by α, that would be the starting point for any external attack. A secondary, reverse encryption will remove this vulnerability.

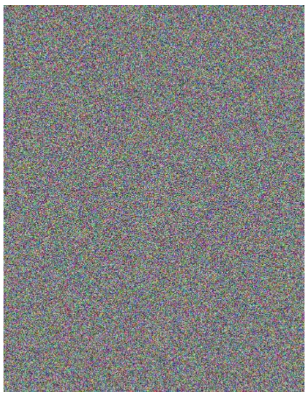
For the decryption process, the exact reverse is carried out. The user will be required to input the value of α, and then the prime store S is generated, and then B is generated, and then the original image P is generated by using the xor operator.

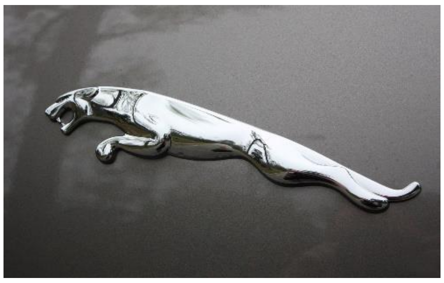
In this manner, a secure encrypted image is generated with no observable pattern to the original, which was the original target.

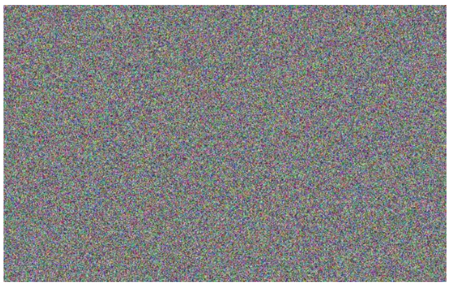
Results:

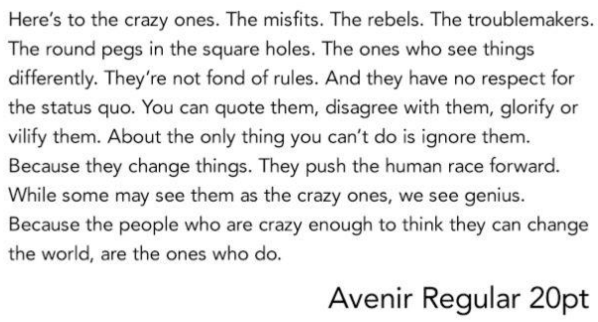
Original image:

Encrypted image:



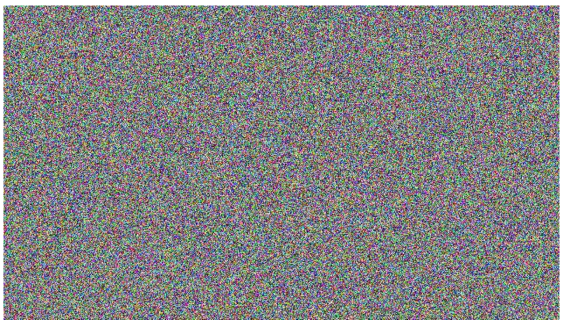






Here's to the crazy ones. The misfits. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They're not fond of rules. And they have no respect for the status quo. You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward. While some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do.

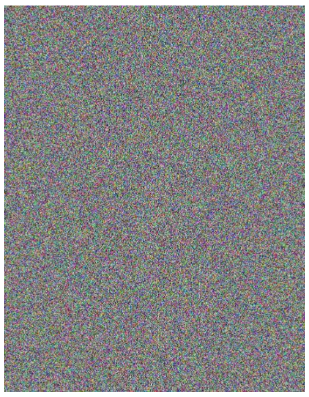
Avenir Regular 20pt

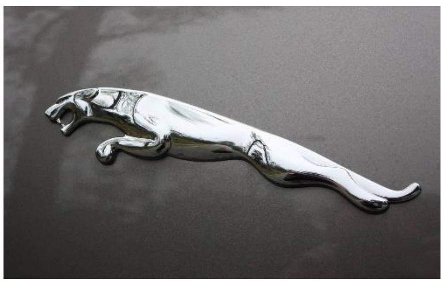


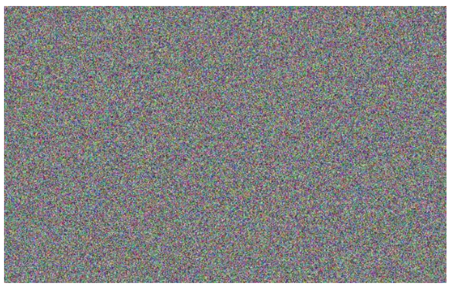
Analysis:

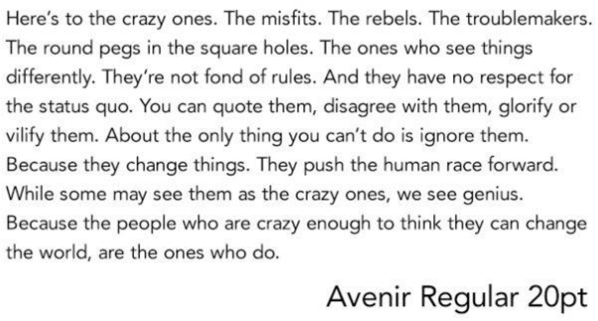
1. Visual test:

Visual observation is an important factor in an image encryption. The lesser the resemblance between the encrypted and original images, the better the encryption scheme. This also implies that no information should be deduced by comparing the original and encrypted images.



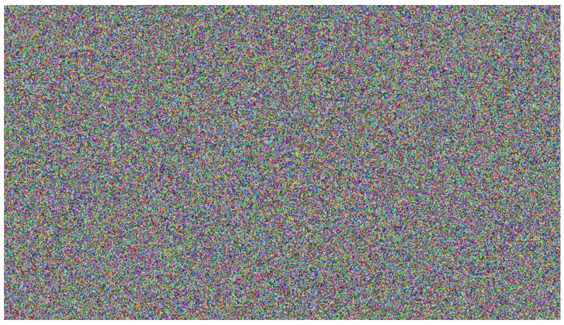






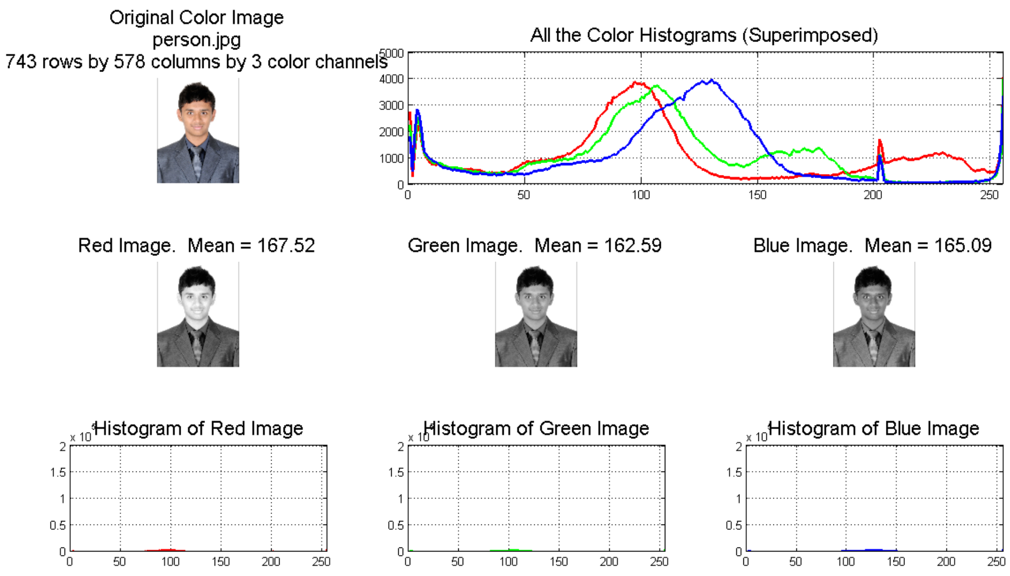
Here's to the crazy ones. The misfits. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They're not fond of rules. And they have no respect for the status quo. You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward. While some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do.

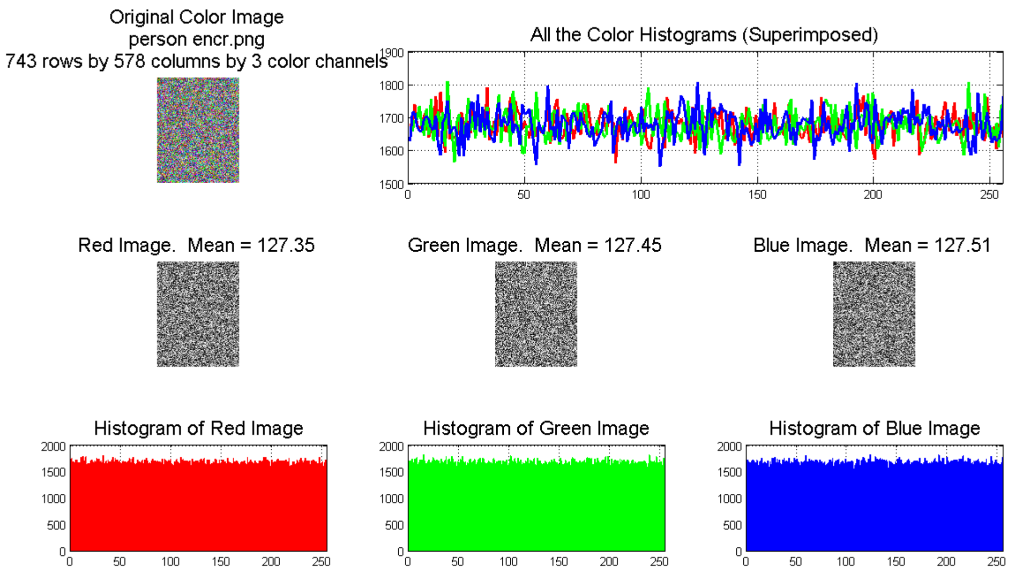
Avenir Regular 20pt

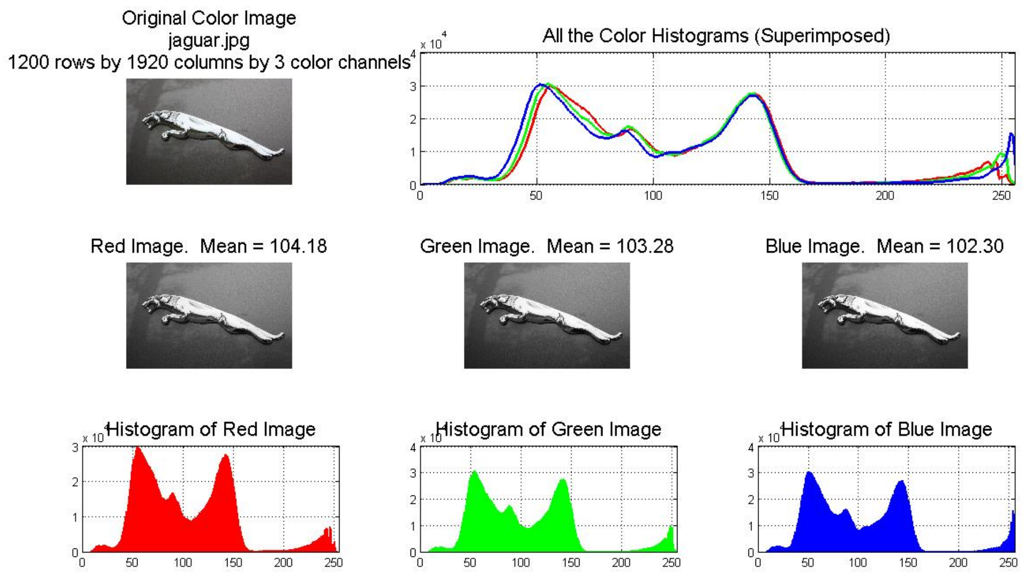


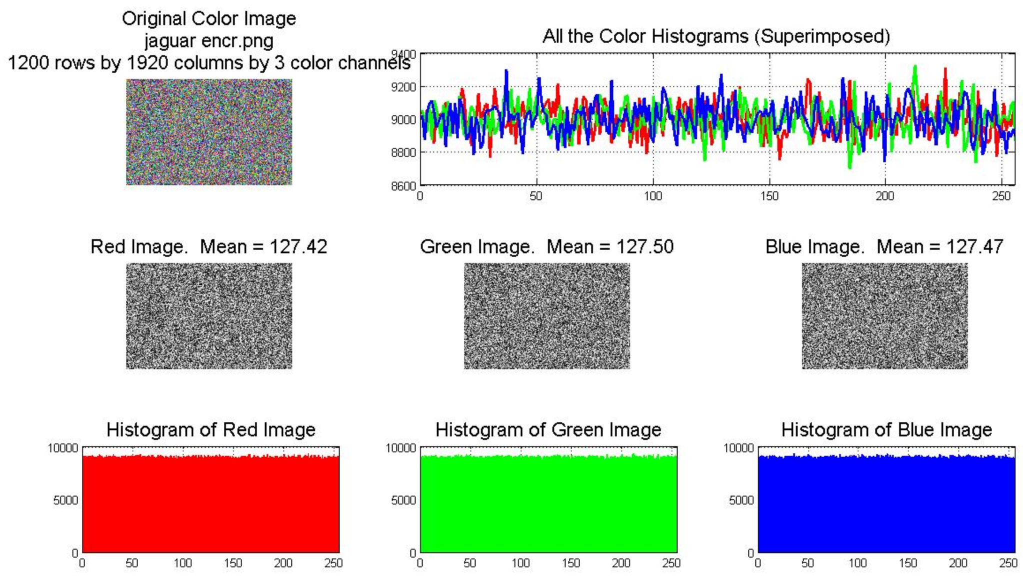
2. Histogram Analysis:

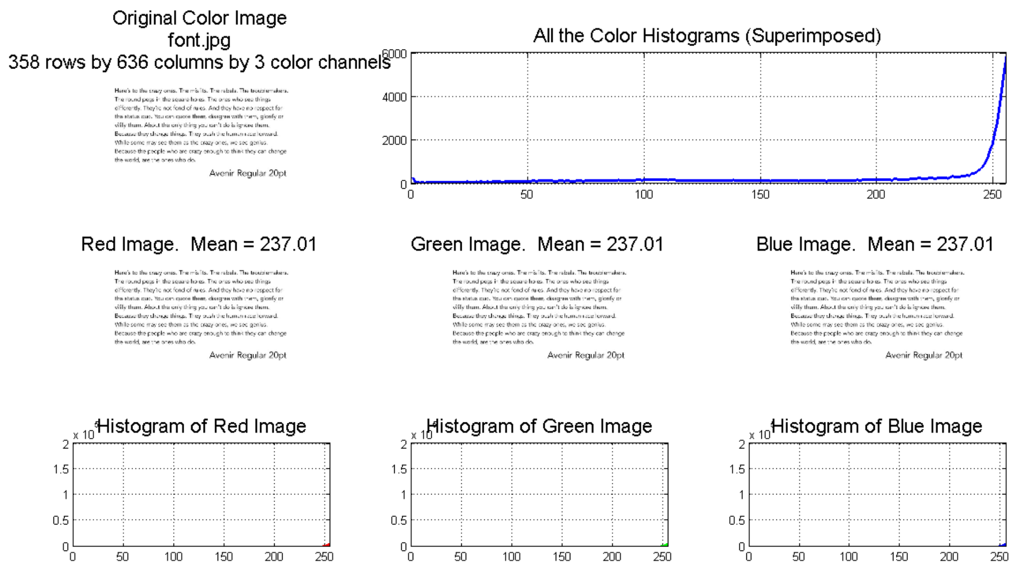
To prevent information leakage from aggressive attacks, it must be ensured that the original and encrypted images do not have any statistical similarity. A histogram analysis expresses the way of the distribution of pixels in the image using the drawing number of observations for each amount of pixels brightness. So, if in the encrypted histogram, if there is an equal probability of generating pixels of all intensities, there is a higher degree of symmetry, as there is no possibility of eliminating attack options. Also, if there is no statistical similarity between the histograms of the original and encrypted images, the attacker with the histogram analysis of the encrypted image cannot acquire information from the original image.

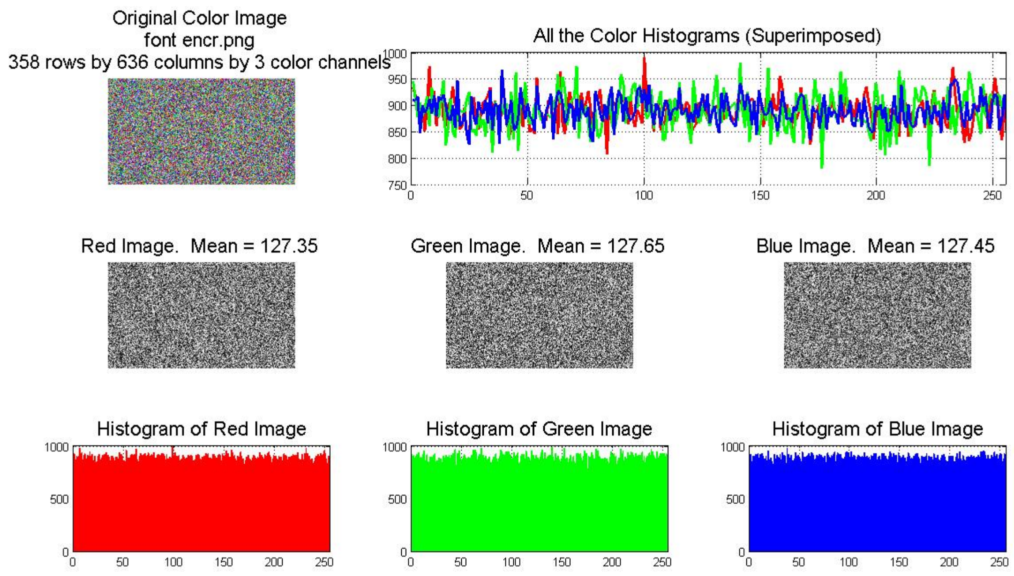





Original Color Image
jaguar.jpg
1200 rows by 1920 columns by 3 color channels
All the Color Histograms (Superimposed)
Red Image. Mean = 104.18
Green Image. Mean = 103.28
Blue Image. Mean = 102.30
Histogram of Red Image
Histogram of Green Image
Histogram of Blue Image


Original Color Image
jaguar encr.png
1200 rows by 1920 columns by 3 color channels
All the Color Histograms (Superimposed)
Red Image. Mean = 127.42
Green Image. Mean = 127.50
Blue Image. Mean = 127.47
Histogram of Red Image
Histogram of Green Image
Histogram of Blue Image





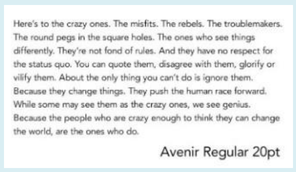
In all three cases, there is no correlation between the histograms of the original and encrypted images. Also, the encrypted histograms show an equal probability for generating R, G, B values of all intensities.

3. Entropy:

Shannon introduced information entropy as the measure of source information in 1949. The entropy of a message source is defined as

$$H(s) = \sum_{i=0}^{2N-1} P(s_i) log_2 \frac{1}{P(s_i)}$$

In this equation, $P(s_i)$ represents the probability of symbol s_i and the entropy is expressed in bits. If we suppose that the source emits 2^8 symbols with equal probability and $s = \{s_1, s_2, s_3, \dots , s_{2^8}\}$, random source entropy is equal to 8. If an encryption algorithm creates symbols with entropy less than 8, there is likelihood to predict original image from encrypted image, which is a threat to the system security. If the value of entropy is closer to ideal value of 8, the encrypted image is more secure.

File	Image	Entropy for proposed algorithm
Person		7.999831211243935
Jaguar logo		7.999968404965706
Avenir Fontface	 Here's to the crazy ones. The misfits. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They're not fond of rules. And they have no respect for the status quo. You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward. While some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do. Avenir Regular 20pt	7.999618633372544

This means that information leakage in the encryption process is negligible and studied algorithms are secure upon the entropy attack.

4. Differential Analysis:

An encryption algorithm should be designed so that it is sensitive to the small changes in the original image. Attacker tries to view the changes result in the encrypted image making minor changes in the original image. Thus, it reveals a significant relationship between the original image and the encrypted image. Also, this action facilitates finding the algorithm key. If a small change in the original image can cause a large change in the encrypted image, then the differential attack is not possible.

Three common measures were used for differential analysis: MAE, NPCR, and UACI. MAE is mean absolute error. NPCR is the number of pixels change rate of encrypted image, while one pixel of original image is changed.

UACI is the unified average changing intensity, which measures the average intensity of the differences between the original image and the encrypted image.

If $C(X,Y)$ and $P(X,Y)$ are the gray level of the pixels at the *x*th row and *y*th column of a W x S encrypted and original image, respectively, then MAE is defined as

$$MAE = \frac{1}{H \times W} \sum_{X=0}^{H-1} \sum_{Y=0}^{W-1} |C(X,Y) - P(X,Y)|$$

Consider two encrypted images C_K and $\overline{C_K}$ that, corresponding to original images, are only different in a pixel. The NPCR is defined as

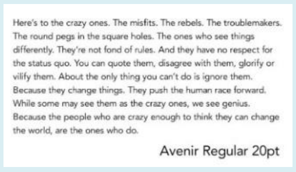
$$NPCR_K = \frac{1}{H \times W} \sum_{X=0}^{H-1} \sum_{Y=0}^{W-1} D_K(X,Y)$$

$$D_K(X,Y) = \begin{cases} 0, & C_K(X,Y) = \overline{C_K}(X,Y) \\ 1, & C_K(X,Y) \neq \overline{C_K}(X,Y) \end{cases}$$

and UACI is defined as

$$UACI = \frac{1}{H \times W} \sum_{X=0}^{H-1} \sum_{Y=0}^{W-1} \left[\frac{|C_K(X,Y) - \overline{C_K}(X,Y)\,|}{255} \right]$$

It is clear that large amounts of NPCR and UACI indicate a high sensitivity of the encryption algorithm to the original image.

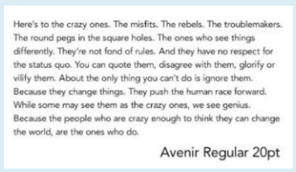
File	Image	UACI for proposed algorithm	NPCR for proposed algorithm
Person		0.303680176844707	0.996077189485564
Jaguar logo		0.182193287037037	0.996084780092593
Avenir Fontface	 Here's to the crazy ones. The misfits. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They're not fond of rules. And they have no respect for the status quo. You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward. While some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do. Avenir Regular 20pt	0.456967429113524	0.996012086715154

The results indicate that the UACI values range from nearly 15% to 50%, and the NPCR values are above 99%. This indicates that the proposed encryption algorithm is highly sensitive to changes in the original image.

5. Correlation Coefficient:

The encrypted image and its original form must bear minimal resemblance for the encryption algorithm to be more robust. If there is little resemblance, then it is harder to find a mapping algorithm from the original image to its encrypted form. The correlation coefficient gives an indication of precisely that. A value closer to one indicates higher resemblance between the original and encrypted forms, and a value closer to 0 indicates lower resemblance.

In statistics, a perfect negative correlation is represented by the value -1.00, while a 0.00 indicates no correlation and a +1.00 indicates a perfect positive correlation. A positive correlation exists when as one variable decreases, the other variable also decreases and vice versa. A negative correlation exists in a relationship between two variables in which one variable increases as the other decreases, and vice versa. A perfect negative correlation means that the relationship that appears to exist between two variables is negative 100% of the time.

File	Image (in grayscale form)	Correlation coefficient for proposed algorithm
Person		-0.002278033631823
Jaguar logo		2.817238980763475e-04
Avenir Fontface	 Here's to the crazy ones. The misfits. The rebels. The troublemakers. The round pegs in the square holes. The ones who see things differently. They're not fond of rules. And they have no respect for the status quo. You can quote them, disagree with them, glorify or vilify them. About the only thing you can't do is ignore them. Because they change things. They push the human race forward. While some may see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do. Avenir Regular 20pt	0.001424567625362

Here, all the correlation coefficients are tending to approach the ideal value of 0, which implies that there is no correlation between the original image and the encrypted image.

Conclusion:

In this manner, a new encryption algorithm has been proposed which uses a key to generate a list of random primes, and consecutively using every list to generate another list. This procedure ensures singularity in the encryption mapping schematics. It has been tested by several techniques, and the entropy tests show results as high as the industry standards. The correlation tests show almost no similarity between the original and encrypted images, and the histogram analysis verifies this assessment. We have carried out visual and statistical tests, and in conclusion, the results show that this algorithm can be expected to find many applications in the real world.