Modify Blowfish Algorithm by Increase the Randomness for S -Boxes

Assist. Prof. Dr. Hala Bahjat Abdul wahab* Lec. Dalia Nabeel Kamal

Abstract

Blowfish weak keys produce "bad" S-boxes, since Blowfish's S-boxes are key-dependent. There is a chosen plaintext attack against a reduced-round variant of Blowfish that is made easier by the use of weak keys. In this paper proposed new algorithm to increase the robust of blowfish algorithm and solve the weakness of sub key problem by generate strong a cryptographic randomness keys((independent keys) and used in Blowfish's S-boxes stages. The proposed algorithm helps to reduce time executions for the algorithm and appropriated for application in which the secret key changes frequently by using randomness digital image that have very big randomness keys space feature .

Keywords: applied cryptography, blowfish algorithm, graphics, image generation, and randomness tests.

تحسين خوارزمية Blowfish من خلال زيادة عشوائية S-Boxes

الخلاصة

خوارزمية Blowfish تولد مفاتيح ضعيفة مما يقود إلى توليد S-boxes سيئة طالما هذه المفاتيح تحوي على مقدار كبير من الارتباط, مما تعمل على مساعدة المهاجم في مهاجمة هذه الخوارزمية بطريقة اختيار النص الصريح للمهاجمة. البحث يطرح خوارزمية جديده تعمل على تحسين خوارزمية Blowfish من خلال حل مشكلة المفاتيح الضعيفة عن طريق الاعتماد على مفاتيح ذات درجه عاليه من العشوائية (تقليص درجة الارتباط بين المفاتيح) , واستخدامها كبديل عن المفاتيح ذات الارتباط العالي فيS-boxes و كذلك الخوارزمية المقترحه ساعدت في تقليل من وقت التنفيذ العالي للخوارزمية وجعلها اكثر مرونه وامكانية استخدامها في التطبيقات التي تتطلب تغيير المفتاح السري بشكل متواصل وذلك بسبب كبر عدد المفاتيح العشوائية التي توفرها الطريقة المقترحه من خلال الاعتماد على توليد صور رقيمة ذات عشوائية عاليه وباحجام مختلفة.

*Computer Science Department, University of Technology/ Baghdad Hala_bahjat@yahoo.com
Daliank_comp@yahoo.com

Introduction

Blowfish is a symmetric block cipher that can be effectively used for encryption and safeguarding of data. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for securing data. Bruce Schneier as a fast, free alternative to existing encryption algorithms designed blowfish in 1993. Blowfish is unpatented and license-free, and is available free for all uses. Blowfish is a variable-length key block cipher. It is suitable for applications where the key does not change often, like a communications link or an automatic file encrypt or. It is significantly faster than most encryption algorithms when implemented on 32-bit microprocessors with large data caches [1].

In cryptography, a weak key is a key which when used with a specific cipher, makes the cipher behave in some undesirable way. Weak keys usually represent a very small fraction of the overall key space, which usually means that if one generates a random key to encrypt a message weak keys are very unlikely to give rise to a security problem. Nevertheless, it is considered desirable for a cipher to have no weak keys. A cipher with no weak keys is said to have a "flat", or "linear", key space. Blowfish's weak keys produce "bad" S-boxes, since Blowfish's S-boxes are key-dependent. There is a chosen plaintext attack against a reduced-round variant of Blowfish that is made easier by the use of weak keys. In this paper produce new technique using, generate digital randomness image use to solve the weak keys for Blowfish's algorithm and reduce time executions by using randomness digital image [2].

Description of the Blowfish algorithm [3]

Blowfish is a variable-length key, 64-bit block cipher. The algorithm consists of two parts shown in figure (1): a key-expansion part and a data-encryption part. Key expansion converts a key of at most 448 bits into several sub key arrays totaling 4168 bytes.

Data encryption occurs via a 16-round Feistel network. Each round consists of a key dependent permutation, and a key- and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.

Sub keys

Blowfish uses a large number of sub keys. These keys must be pre-computed before any data encryption or decryption.

The P-array consists of 18 32-bit sub keys:

P1, P2,..., P18.

There are four 32-bit S-boxes with 256 entries each:

S1, 0, S1, 1,..., S1,255;
S2, 0, S2, 1,..,, S2,255;
S3, 0, S3, 1,..., S3,255;
S4, 0, S4, 1,..,, S4,255.

Encryption process

Blowfish has 16 rounds.

The input is a 64-bit data element, x.

Divide x into two 32-bit halves: xL, xR.

Then, for i = 1 to 16:

xL = xL XOR Pi
xR = F(xL) XOR xR
Swap xL and xR

After the sixteenth round, swap xL and xR again to undo the last swap.

Then, xR = xR XOR P17 and xL = xL XOR P18.

Finally, recombine xL and xR to get the cipher text.

Decryption is exactly the same as encryption, except that P1, P2,..., P18 are used in the reverse order. Implementations of *Blowfish* that require the fastest speeds should unroll the loop and ensure that all sub keys are stored in cache.

Generating the Sub keys

The sub keys are calculated using the *Blowfish* algorithm [3]:

1. Initialize first the P-array and then the four S-boxes, in order, with a fixed string.

This string consists of the hexadecimal digits of pi (less the initial 3):

P1 = 0x243f6a88, P2 = 0x85a308d3, P3 = 0x13198a2e, P4 = 0x03707344, etc.

2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XOR-ed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.).
3. Encrypt the all-zero string with the *Blowfish* algorithm, using the sub keys described in steps (1) and (2).
4. Replace P1 and P2 with the output of step (3).
5. Encrypt the output of step (3) using the *Blowfish* algorithm with the modified sub keys.
6. Replace P3 and P4 with the output of step (5).
7. Continue the process, replacing all entries of the P array, and then all four S-boxes in order, with the output of the continuously changing *Blowfish* algorithm. In total, 521 iterations are required to generate all required sub keys. Applications can store the sub keys rather than execute this derivation process multiple times.

Blowfish algorithms with weak keys [2]

Blowfish weak keys produce "bad" S-boxes, since Blowfish's S-boxes are key-dependent. There is a chosen plaintext attack against a reduced-round variant of Blowfish that is made easier by the use of weak keys. This is not a concern for full 16-round Blowfish. The sub key-generation algorithm does not assume that the key bits are random. Even highly correlated key bits, such as an alphanumeric ASCII string with the bit of every byte set to 0, will produce random sub keys. However, to produce sub keys with the same entropy, a longer alphanumeric key is required [3]. The math behind Blowfish consists of 16 rounds, or loops. Cryptanalysis of Blowfish by Serge Vaudenay reveals a partial differential attack that can recover the plaintext array in 28r+1 chosen plaintexts. There is also a class of known weak keys that can increase the effectiveness of this attack by a two facts.

Hash Visualization Technique [5]

Researchers have been trying to make cryptographic primitives stronger against attacks. Contribution is to propose the new security primitive hash visualization, to establish the necessary requirements, Random Art as a prototypical solution, and finally, to show how to apply hash visualization to improve the security of roots key validation and user authentication.

Since Random Art is just a prototype of the Final solution, we hope with this paper to direct the interest of researchers in image processing, security, and psychology and cooperation between them in order to find better solutions.

Simple Generated Digitl Images [6]

The final image is established by combining two picture shown in figure (6), pattern and back ground generated as follows:

- **First:** One represents the background shown in figure (3),with a size of 256-256 pixels, pattern and background drawn by repeating segment (i.e draw lines)with different value and different color in each time.
- **Second**: One represent the patterns drawing by using (the graphical curves sin, cos, circil, curve equation like bizer oespline)shown in figure (4), adjustment by different values limited

by maximum value depeds on boundaries of background and segment shape .

This technique gives opportunity to change the shape or color in an easy way to get different new patterns and then diferent pictures in any time with high random color(value) in each small area in the picture. When compare this technique with the nomal color image we see in each small area the pixels have the nearest values. Two images are shown in figure-(5),where the first one is a normal ture color image(the apple) ,while the second one is a generated image drawn using a circle equation. Arandom values of RGB to each pixel is also shown in figure-(5). It is clear from the nearest value from the first image and the randomness values from the second image[5].

Generate Randomnes Digital Images using Random Mathematical Formula

The following illustrates the proposed algorithm to generate randomness digital image by apply Random mathematical formula to red, green and blue seed , on digital image that generate according pervious secetion-4(Simple Generated Digitl Images), example for the formules[7].

Red=cos(cos(arctan(cos(tan(mult(sin(tan(arctan(mult(mult(rgb(y[],r,x[ ]),(rgb(r,x[],x[ ])),bw(y[])))))),bw(y[])))))).

Green=arctan(mult(cos(arctan(tan(mult(cos(tan(tan(arctan(mult(mult(tan(y[ ]),arctan(r)),cos(r))))), r)))), cos(y[ ]))).

Blue= if(y[],x[],mult(cos(mod(mix(if(reverse(exp(mod(div(sin(mix(sin(r),sin(x[]),mix(y[],y[],r,y[]),bw(r))),bw(x[ ])),rgb(y[],y[],r)))),y [],add(x[],r)),y[],x[ ],add(x[ ],x[])),add(y[ ],y[ ])),bw(y[]))).

The ***BW function*** is a user-defined function,which determines the gray color of input value and it can be calcuted using the equation is defined as follows[8,9]:

G=trunc((R*0.30)+(G*0.11)+(B*0.59))

The ***revers function*** is a user-defined function,which determines the revers color of the input value and it can be calculated using the equation is defined as follows: **R=255-X**

Where x is the input value to the reverse functionand'255' is the upper rangeof the colors range, whichis between 0 and 255.

The *mix function* is user-defined function, which determines the mixing of two colors'a'and'b' depending on the value of 'c' and 'd'. it can be calculated using the equation is defined as follows: **M=(a*c)+(b*d)/(a+b)**

Where a,b,c,d are the input color values. Each formula is evaluated to produce one component value for each pixel (x,y). the three-color-component values define the R-value,G-value and b-value, they are mixed to produce the final RGBvalue for each pixel.

Algorithm (1): proposed algorithm to generate randomness digital image.

Input: generate digital image that generate according section-4:

Output: Obtain to randomness image.

Process:

Step1: open the generated image and perform the following process to separate the color pixels (RGB) to red, green and blue and save colors(R, G, B) in one-dimension array (pic1)

```
For i = 0 to height
For j = 0 to width
pixel = Picture1.Point(i, j)
R = pixel& Mod 256
pic1 (k1) = r
k1 = k1 + 1
G = ((pixel& and &HFF00FF00) / 256&) Mod 256& pic1(k1)
Pic1 (k1) =g
```

```
B = (pixel& and &HFF0000) / 65536
k1 = k1 + 1
pic1 (k1) = b
 k1 = k1 + 1
Next j
     Next i
```

Step2: *(generate image using Random mathematical formulas)*

```
K2=0

For i = 0 to height
          For j = 0 to width

Red=cos(cos(arctan(cos(tan(mult(sin(tan
 (arctan(mult(mult(rgb(y[ pic1(i )
 ],r,x[pic1(i)),(rgb(r,x[pic1(i )],x[pic1(i
 )])),bw(y[pic1(i )])))),bw(y[pic1(i
 )])))))).

  Pic2(k2)=red

  K2=K2+1

Green=arctan(mult(cos(arctan(tan(mult(c
 os(tan(tan(arctan(mult(mult
 (tan(y[pic1(i )]),arctan( r) ),cos( r))))),
 r)))), cos(y[pic1(i )]))).

 Pic2(k2)=green

  K2=K2+1

Blue=f(y[i],x[i],mult(cos(mod(mix(if(rev
 erse(exp(mod(div(sin(mix(sin( r),sin(x
 [i]),mix(y [i],y[i],r,y[i]),bw(r) )),bw(x[i
 ])),rgb(y[i],y[i
 ],r)))),y[i],add(x[i],r)),y[j],x[i],add(x[i],
 x[i])),add(y[i],y[i]))),bw(y[i]))).

Pic2(k2)=blue

 K2=k2+1

Next j ,Next i
```

Step3: (XOR image by image)

Perform XOR operation directly on the two images pic1(image generate by simple method sec-4) and pic2 (image generate according *Random mathematical formulas)* and save the result in third picture (pic3) as follow:

```
  Set   k3 = 0
For i = o to k1 - 1
  pic3 (k3) = pic1 (i) XOR pic2 (i)
   k3 = k3 + 1
    Next i
```

Step4*: (randomness digital image)*

Separate the three colors (R, G, B), save the three one- dimension arrays red, green and blue, and then use the function (RGB) to obtain the true color pixels. This process performs as follow:

```
    Set i = 0, k1 = 0, k2 = 0, k2 = k3 / 3
 Do While i < k2
   red (i) = pic3(k1)
   k1 = k1 + 1
  green (i) = pic3(k1)
  k1 = k1 + 1
  blue (i) = pic3(k1)
  k1 = k1 + 1
  i = i + 1
 loop
  k1 = 0
 For i = 0 to height
 For j = 0 to width
    a (i, j) = RGB(red(k1), green(k1),
blue(k1))
 Picture3.PSet (i, j), a (i, j)
 k1 = k1 + 1
 Next j
  Next i
```

Step5: end.

In the following figure (7) shown the block diagrams for proposed algorithm for generate randomness image.

Test the randomness for generates images [9]

The proposed methods to generate randomness digital give good results according the randomness test for digital images. Table (1): shown the test results for randomness images that obtained according the proposed algorithm.

The image tests give the following results:

a-The large resuls of MSE means the proposed method is succeeded to conceal image information.

b- The small results of SNR and PSNR means the proposed method caused large noise (i.e. small, a result implies better image concealment of original image.).

c- The similarity measure shows the amount of correlation between the original image and randomness image and the result from this test is acceptable.

Tests Randomness for sub keys

Different sizes of keys are used in these tests and the results of the tests proved that the keys have the randomness property and can be used as randomness key in cryptography field [10].

Implement the randomness Tests to the two Keys clipped from generated images.

Key 1 (size 130 bits) from image1:

1- **Frequency test:** 0.277 **Pass value** Must be ≤ 3.84

2- Run test: **Pass value** Must be ≤ 22.362

- $\mathbf{T_{0:}}$ 3.638 $\mathbf{T_{1:}}$ 4.638

3- Poker test: 4.585 **Pass value** Must be ≤ 11.1

4- Serial test: 2.969 **Pass value** Must be ≤5.99

5- Auto Correlation test for ten bits:

Pass value Must be ≤3.48

Shift 1	1.310
Shift 2	0.281
Shift 3	0.386
Shift 4	2.571
Shift 5	0.392
Shift 6	0.516
Shift 7	0.008
Shift 8	0.295
Shift 9	1.860
Shift 10	2.700

Key 2 (size 520 bits) from image1:

1-**Frequency test** 1.025

Pass value Must be ≤ 3.84

2-**Run test:**

Pass value Must be ≤ 22.362

- $\mathbf{T_{0:}}$ 0.338 - $\mathbf{T_{1:}}$ 15.038

3- Poker test: 6.862

Pass value Must be ≤ 11.1

4- Serial test: 1.723

Pass value Must be ≤5.99

5- Auto Correlation test for ten bits:

Pass value Must be ≤3.48

Shift 1	1.10
Shift 2	0.124
Shift 3	0.095
Shift 4	1.116
Shift 5	0.049
Shift 6	0.070
Shift 7	1.639
Shift 8	2.133
Shift 9	0.440
Shift 10	0.635

The Proposed methode to solve the weak subkey for Blofish algorithm

The proposed method is modifying the sub keys calculated using the modify *Blowfish* algorithm:

1. Initialize first the P-array and then the four S-boxes, in order, by clipping pixels value (R,G,B)from randomness digital image according some fixed schema (by use mathematical equation like sine, cos , sine –curve… etc , see [5]) ,after pass all randomness tests and pass the main two conditions:

A- Checking generated keys and rejection of weak keys into the key scheduling.

B- When the number of weak keys is known to be very small (in comparison to the size of the key space), generating a key uniformly at random ensures that the probability of it being weak is a (known) very small number.

2. using pic3 (see alg.2). Pic3 (1) = color pixel (Red), Pic3 (2) = color pixel (Green), Pic3 (3) = color pixel (Blue), etc.
3. Encrypt the initial plain text instead of all-zero string using the sub keys described in steps (1).
3. Replace P1 and P2 with the output of step (3).
4. Encrypt the output of step (3) using the *Blowfish* algorithm with the modified Sub keys.

5. Replace P3 and P4 with the output of step (5).
6. Continue the process, replacing all entries of the P array, and then all four S-boxes in order, with the output of the continuously changing *Blowfish* algorithm.

In total, 521 iterations are required to generate all required sub keys. Applications can store the sub keys rather than execute this derivation process multiple times.

Conclusion

The paper studied the blowfish weakness. The most important of them are:

- S-boxes weakness based on collision.
- Some vulnerability in steps of the key generation process.

In this research, we tend to overcome this weakness by making some modification on generation of sub keys:

1. Modify P-array and four S-boxes, by clipping pixels value (R,G,B)from randomness digital image according some fixed schema (like use mathematical equation like sine, cos, sine –curve… etc) ,after pass all randomness tests.
2. Then generate the final sub keys by encryption the initial plain text instead of all-zero string.
3. The generated randomness digital image increases the key space , the computation time reduce and reach to randomness sub keys from the attacker become more difficult.
4. The proposed algorithm it is suitable for applications where the key does change often, like a communications link, because The generated randomness digital image increases the key space and clipped may randomness sub key from the sane image by using different schema each time.

References

[1] ***"Encryption using the blowfish algorithm",*** www.cryptosys.net & www.dimgt. com.au, 2002.

[2] "***The blowfish encryption algorithm—one year last***", www.schneier.Com personal web site.

[3] ***"Description of a new variable-length key, 64-B cipher (blowfish)",*** www.fincrypt@fincrypt.net, 2002.

[4] ***"On the weak keys of blowfish",*** vaudenay@dmi.ens.f, 1995.

[5] *Hilal M. Y., l Monem. R, Hala. B. (2007),"* ***Using Generated Digital Images to Modify the PGP Cryptography Protocol",*** SAM'07, *International Conference on Security and Management.*

[6] Nada Al-ubaidy, (2002)." ***Cipher by Image Processing***". M. Sc. Thesis of Military College of Engineering, Computer Science Department.

[7] Akhlas Abbas Al Bahrany, (2001***)." Structured Image Authentication System"***, M.Sc. Thesis, University of technology at Computer Science.

[8] Dhamija Rachna, (2000). "***Hash visualization in user authentication"***. In proceedings of the computer Human Interaction Conference.

[9]Andrej Bauer. Gallery of random .http://www.cs.cmu.edu/~andrej/art/, 1998.

[10] William Stalings,(2003). "***Cryptograhy and Network Security, (Principles and Practice)***", Pearson Education, Inc.

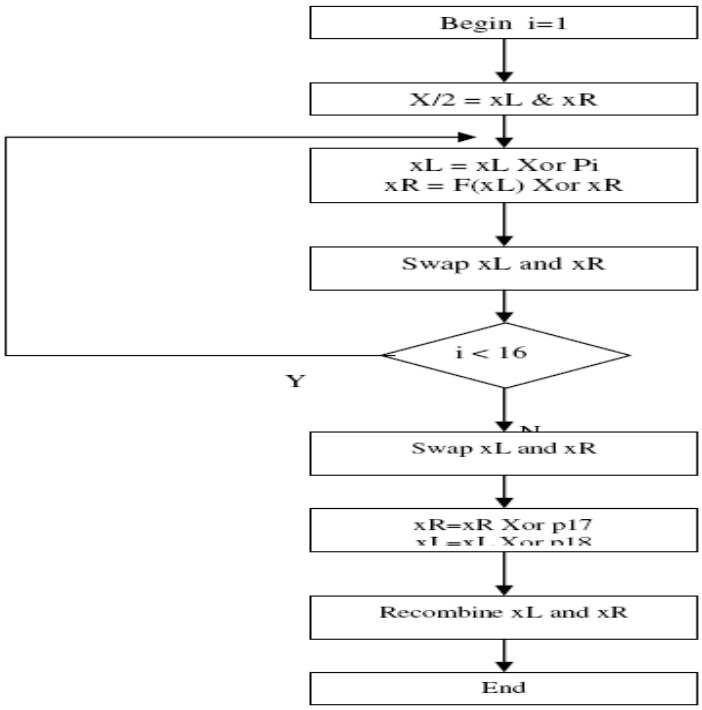


Figure 1. Blowfish algorithm diagram

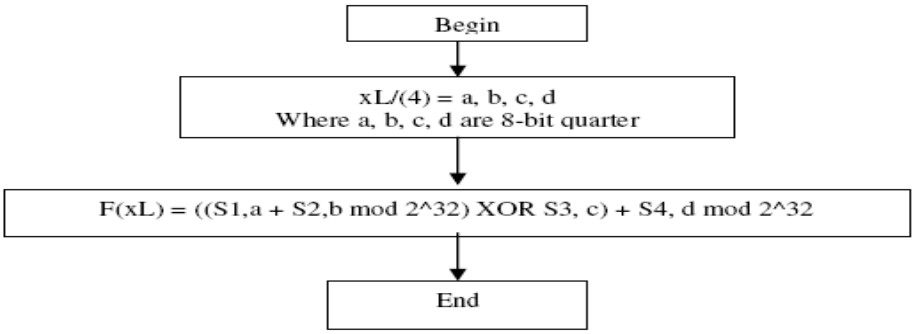


Figure (2)-(F)-Function algorithm diagram

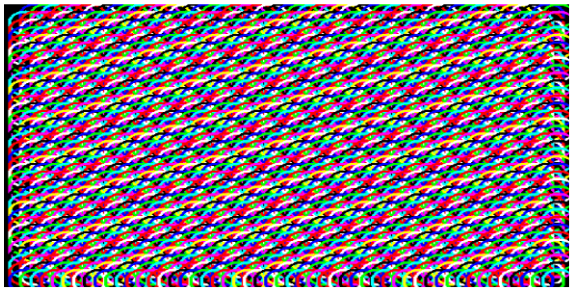


Figure 3. Pic1 : represent background for generated image consist from recrsive circles with different color for each circle.

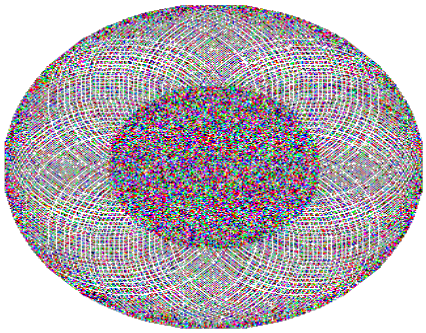
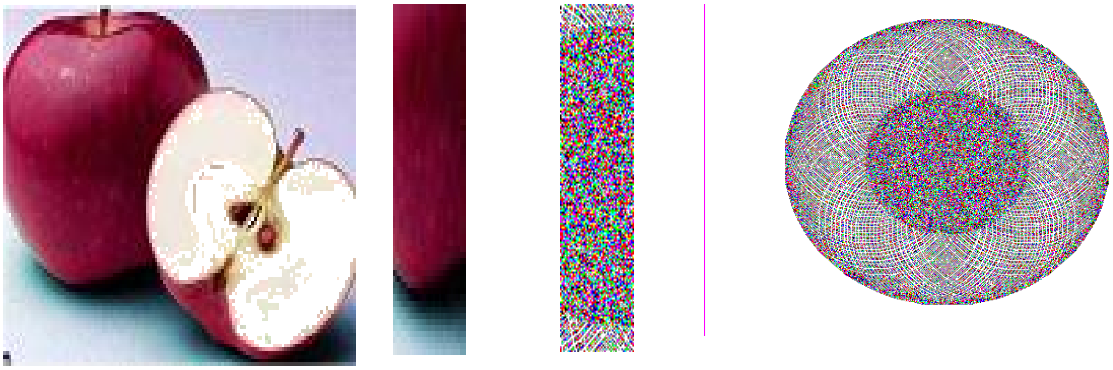


Figure 4. pic2: represent pattren frogenerated image which consis from drwing recursive circles according circle equation with different color for each pixel in each circle



5	184	198
210	255	255
163	192	190
123	74	121
250	229	225
20	3	22
237	255	255
131	198	167
61	129	106
0	27	45

RGB1

112	95	69
117	97	70
48	125	93
148	125	93
145	120	89
155	130	100
153	129	105
159	137	114
158	138	114
115	99	76

RGB2

Figure 5. Comparesim between normal color image and generaterd image

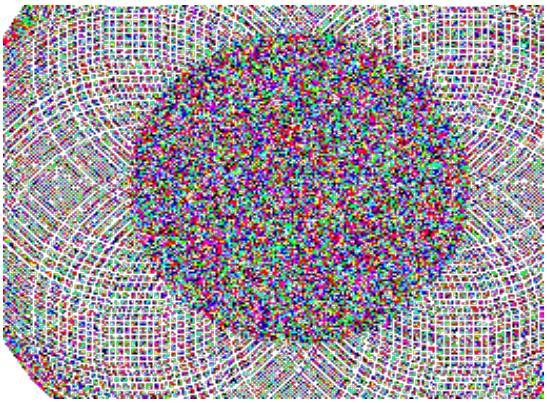


Image1: using circle equation.

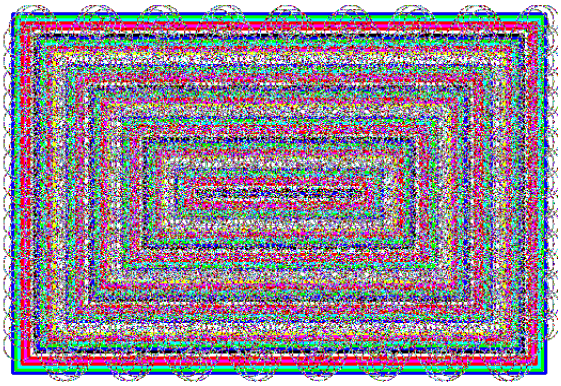


Image2:using regtangles and cos equation.

Figure 6. Generated Digitl Image

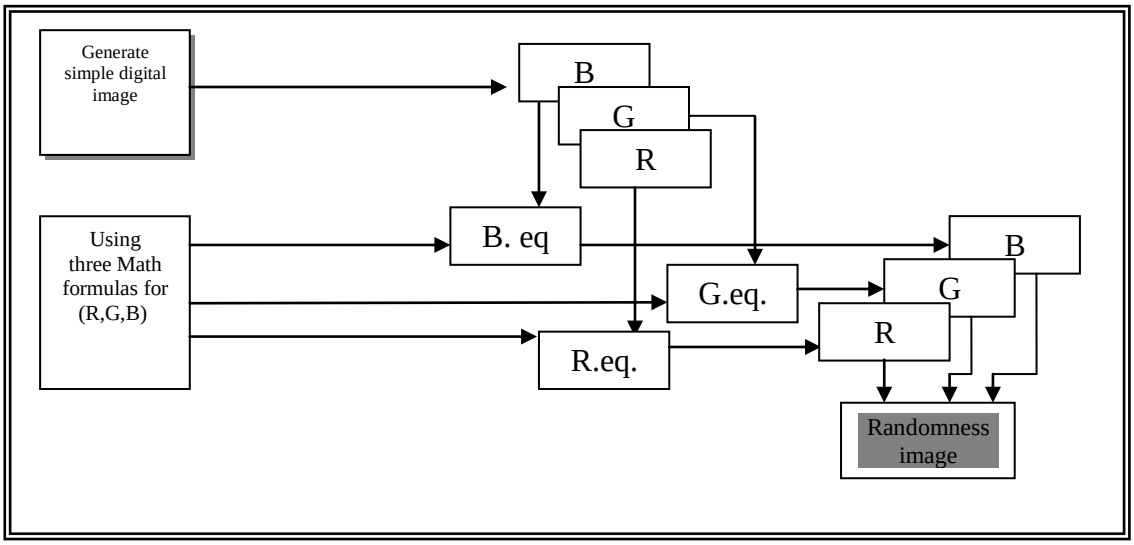


Figure 7. block diagrams for proposed algorithm for generate randomness image.

Example of the* results *proposed algorithm:

The following examples of images show the results of implementing the pervious algorithm (1).

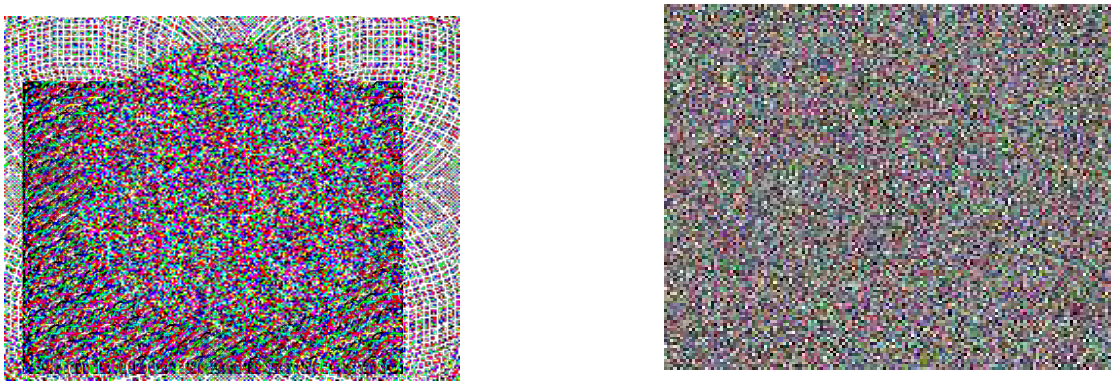


Image1

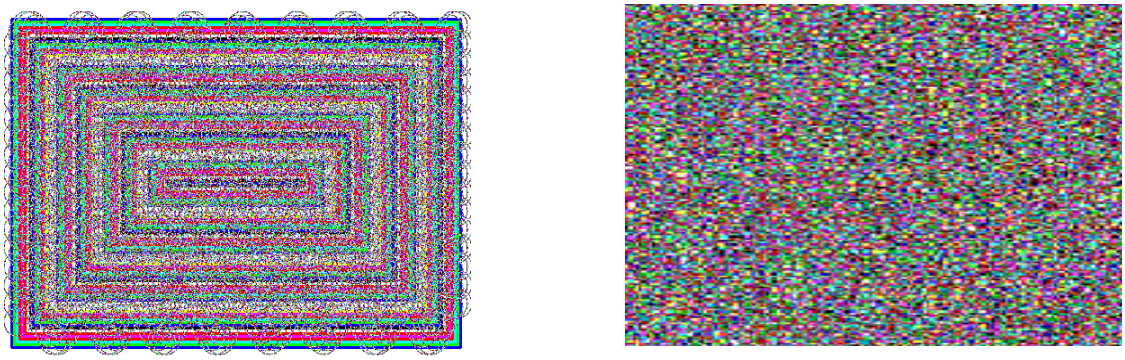


Image2

Generate Simple digital image (see sec-4) ***Randomness digital images after perform mathematical formal***

Figure 8. *the Generated images*

Table 1. shown the test results for randomness images.

Image Name	MSE			PSNR			SNR			Similarity		
	R	G	B	R	G	B	R	G	B	R	G	B
Image1	10243.63	13398.25	4193.79	1.68	1.22	2	5.46	2.66	1.81	0.20	0.11	0.31
Image2	12476.29	12123.87	3226.05	1.68	1.44	1	5.00	4.11	3.70	0.18	0.16	0.51