Eng. & Tech. Journal, Vol. 28, No. 2, 2010

Generate Random Image-Key using Hash Technique

Dr. Nidaa F. Hassan* Akbas E. Ali & Teaba Wala Aldeen*

Received on: 2 / 3 /2009
Accepted on: 1/ 10 /2009

Abstract

Random image is very useful to be used as a source for clipping randomness session keys in cryptography. In this paper a new algorithm is proposed to generate random image .The random image is generated by mixing initialization vector with normal digital image; the mixing process is applied in the same way as HASH technique. A special cryptography algorithm for generating random numbers is used to generate initialization vector. This proposed algorithm is capable of generating random image that can meet security requirements of cryptographic algorithms.

Keywords: cryptography, random image in cryptography, hash function, random key.

توليـد الصـور العشـوائيه(مفتاح) بأسـتخدام تقنيـه المـزج

الخلاصه

الصوره العشوائيه مفيده جدا حيث يتم استخدامها كمصدر للحصول على مفـاتيح دوريـه عشوائيه في التشفير . في هذا البحث تم اقتراح خوارزميه جديده لتوليد صوره عشـوائيه . ان الصوره العشوائيه يتم توليدها بمزج متجه ابتدائي مع صوره رقميه اعتياديه , عمليه الخلـط تطبق على غرار تقنيه المزج . خوارزميه خاصه لتوليد الارقام العشوائيه قد تم استخدامها لتوليد المتجه الابتدائي . ان الخوارزميه المقترحه قادره على توليد صـوره عشـوائيه التـي تلبـي المتطلبات الأمنيه لخوارزميات التشفير.

1. Introduction

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication, and data origin authentication [1].

The security of encrypted data is entirely dependent on two things: the strength of the cryptographic algorithm and secrecy of the key [2].

In cryptography, a key is a piece of information (a parameter) that determines the functional output of a cryptographic algorithm or cipher. Without a key, the algorithm Would have no result. In encryption, a key specifies the particular Transformation of plain text into cipher text, or vice versa during decryption. Keys are also used in other cryptographic algorithms, such as digital signature schemes and message authentication codes [3].

To prevent a key from being guessed, keys need to be generated truly randomly and contain sufficient entropy. Since random number is one whose value cannot be predicted, computers are not very good at producing truly random data. Instead,

*Computer Science Department,University of Technology/Baghdad,Iraq

https://doi.org/10.30684/etj.28.2.15
2412-0758/University of Technology-Iraq, Baghdad, Iraq

they rely on a pseudo-random number generator (PRNG), so strong cryptographically must seeded with truly random values [4].

The problem of how to safely generate truly random keys is difficult, and has been addressed in many ways by various cryptographic systems. In this paper a new method is proposed to generate random key which can be clipped from random image.

2. Basic Cryptographic Technologies

There are several ways of classifying cryptographic technologies. For purposes of this paper, they will be categorized based on the number of keys that are employed for encryption and decryption. The three types of algorithms are shown in (Figure 1):

A. Secret Key Cryptography (SKC): Uses a single key for both encryption and decryption [5]. Secret key cryptography assumes that the parties who share a key rely upon each other not to disclose the key and protect it against modification. The best known secret key system is the Data Encryption Standard (DES), published by NIST as Federal Information Processing Standard (FIPS) 46-1 [6].

B. Public Key Cryptography (PKC): Uses one key for encryption and another for decryption [5]. Whereas secret key cryptography employs a single key shared by two (or more) parties, public key cryptography uses a pair of keys for each party. One of these is "public" and one "private". The private key must be kept confidential and be known only to its owner. There are several public key cryptographic systems. One of the first public key systems, named RSA after its three MIT creators, Ronald Rivest, Adi Shamir, and Len Adleman, is in wide use and can provide many different security services [6].

C. Hash Functions: Uses a mathematical transformation to irreversibly "encrypt" information [5]. One common use of hash functions is to “destroy” any structure that may exist in the input, while preserving most of its entropy. Validity of using hash functions for entropy extraction is not based on their cryptographic properties but rather on our belief that a good hash function destroys most of the dependencies that may exist in the bits of its input [7].

2.1. Cryptographic Hash Function

Cryptographic Hash functions take a message as input and produce an output referred to as a hash code, hash-result, hash-value, or simply hash. More precisely, a hash function h maps bit strings of arbitrary finite length to strings of fixed length, say n bits. For a domain D and range R with h: D—›R and |D| >|R|, the function is many-to-one, implying that the existence of collisions (pairs of inputs with identical output) is unavoidable. Indeed, restricting h to a domain of t-bit inputs (t > n), if h were “random” in the sense that all outputs were essentially equiprobable, then about 2t−n inputs would map to each output, and two randomly chosen inputs would yield the same output with probability 2−n (independent of t).The basic idea of cryptographic hash functions is that a hash-value serves as a compact representative image (sometimes called an imprint, digital fingerprint, or message digest) of an input string, and can be used as

if it were uniquely identifiable with that string [1].

2.1.1 Basic Properties of Cryptographic Hash Function

To facilitate further definitions, three potential properties are listed (in addition to ease of computation and compression), for an unkeyed hash function h with inputs x, x0 and outputs y, y0.

1. Preimage resistance—for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find any preimage x0 such that h(x0) = y when given any y for which a corresponding input is not known.
2. 2^{nd}-preimage resistance—it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, to find a 2^{nd}-preimage x0 6= x such that h(x) = h(x0).
3. Collision resistance—it is computationally infeasible to find any two distinct inputs x, x0 which hash to the same output, i.e., such that h(x) = h(x0). (Note that here there is free choice of both inputs [1].

2.2.2 Basic Requirements for Cryptographic Hash Function

The basic requirements for a cryptographic hash function are:

1. The input can be of any length,
2. The output has a fixed length,
3. H(x) is relatively easy to compute for any given x,
4. H(x) is one-way,
5. H(x) is collision-free [8].

2.2.3 Hash Function as Compression Function.

A compression function takes a fixed length input and returns a shorter, fixed-length output. Given a compression function, a hash function can be defined by repeated applications of the compression function until the entire message has been processed. In this process, a message of arbitrary length is broken into blocks whose length depends on the compression function, and "padded" (for security reasons) so the size of the message is a multiple of the block size. The blocks are then processed sequentially; taking as input the result of the hash so far and the current message block, with the final output being the hash value for the message (see Figure (2) [8].

4. Randomness in Cryptography

Most cryptographic algorithms are based on a controlled random number generator [9].

For any cipher, it's extremely important that communicating parties choose the key at random, without any possible bias or correlation between bits. In order for a cryptographic algorithm to accomplish its goals, keys must have at least the following properties:

1. Keys must be unguessable, in the sense that any efforts that would enable the key to be discovered must be deemed outside the abilities of the presumed attacker. This implies that a key must be drawn from a distribution with sufficient entropy to render it computationally infeasible to exhaustively search (or otherwise to prevent the attacker from confirming a guess at the key, though this is typically much harder). In addition, the key must remain computationally infeasible to find in light of all the information the attacker is able to gather about it.

2. Keys must be reproducible by intended parties when needed to perform cryptographic encoding has a corresponding decoding action , and since at least one of these two (and often both) require possession of the secret key , typically the key will need to be reproduced spatially or temporally. For example, a key may need to exit at two different computers simultaneously (a spatial reproduction) if they are interacting in a cryptographic communication protocol. A key may need to exit in the same computer at different times (a temporal reproduction), but not in the intervening period, if the key is used to encrypted and decrypt files on the computer [10].

5. The Proposed Method

The proposed method for generating random image is divided in two stages, and they are:

1. Design Random Numbers Generator (to generate initialization vector).
2. Mixing Initial Vector with an Input Image.

1. Design Random Numbers Generator

Random image can be obtained by mixing Initialization vector (IV) with an input image. This initialization vector is a block of random numbers. Random number generator is designed to output random key which can be deal with it as an initialization vector (IV).

Keeping keys secret is one of the most difficult problems in practical cryptography, so to prevent a key from being guessed, keys need to be generated truly randomly and contain sufficient entropy. This stage is divided into following steps:

A. Enter a password and making padding process on it. The purpose of padding process is to make a password a multiple of 128, 256 and 512. The following example specifies how this process is performed:

The password is **"Random Image –Key"**; its length is 17 characters. After padding process (assuming that the required length is 32 characters), the password will be:

"**Random Image –Key;] Ac%Gi+Mo1Su**" .

Algorithm (1) illustrates the padding password to any required length:

B. Convert the padded password to its binary code, so the output will be a stream of bits (zeros and ones).

C. Set the parameter of random generator. three parameters are generated (R,N and S) such that:

1. Parameter R from (0) to (65535), its binary base (2^{16}).
2. Parameter N from (0) to (31), its binary base (2^{5}).
3. Parameter S from (0) o (127), its binary base (2^{7}).

With each different parameter (a typically random stream of bits used to generate a usually longer pseudo-random stream), the pseudo-random number generator generates a different pseudo-random sequence. Algorithm (2) illustrates the generation of three parameters.

N and S parameters are constructed in the same way as R, the only difference is the run time since N is from 0 to 31 and S is from 0 to 127.

D. Construct Sequence Generators, five sequence generators are established in this step, and their values depend on the R, N and S parameters. Algorithm (3)

illustrates the construction of these sequences.

Construction of SQ2 (L2), SQ3 (L3), SQ4 (L4), SQ5 (L5) in same way as SQ1,such that L2=313, L3=231, L4=258 and L5=281.

E. Generating Random Number is accomplished in this step, firstly by specify the size of random key, secondly by arithmetically combined each random number in the five sequence generator using XOR operation.

2. Mixing Initial Vector with Input Image.

Once the steam of random numbers is established from the pervious stage, this stream is considered as initialization vector (IV) of Hash function. (The input image is divided into blocks; each 10 bytes are considered as one block. The (IV) is XOR_ed with first block of input to produce the first block of random image. This cipher block also XOR_ed with second block of input image and the output block which is considered as second block in output random image. This process is run repeatedly to the whole blocks of input images, and the final output is the random image (ciphered image). Algorithm (4) and Figure (3) illustrates this stage.

6. Distortion Measures and Randomness Test Results.

Different random images are generated, since different Initialization vectors (IV) sizes are mixed with the input image. Two test images (Gray and Color) are XOR_ed with different IVs sizes.

Figures (4) and figure (5) are showing the test images, tables (1) and table (2) are showing the constructed random images.

There are large numbers of tests used for evaluating generators. In this section, two test metrics are used which for testing random images, they are:

1. Distortion Measures.

2. Statistical Tests of Randomness.

6.1. Distortion measures.

Distortion measures which are adopted for testing the performance of the suggested method, and they are MAE, MSE, PSNR, SNR and Correlation Measures. These measures define the overall error between (N) samples of the input image Bmp file, and the corresponding samples of the random image (Bmp file). This kind of measures can be a good testing tool for quality, especially concealing the samples of input image. Table (3) lists these distortion measures [11].

Where,

$f(x,y)$ **:** is the value of the original image at row(y) and column(x),

$f'(x,y)$: is the corresponding values of the reconstructed image.

H , W : the height of the image and the width of the image respectively,

A_{ij} ,B_{ij} : two images matrices size (N×N).

Tables (4) and table (5) are showing the results of applying distortion measures on random images (Test image_ 1 and Test_image_2)

From the results listed in the above tables, it is noticed that large MSE gives a good results for concealing the input image samples, while small results of SNR and PSNR a good results of noise occurred to input image. Finally, acceptable results obtained from correlation measure.

6.2. Statistical Tests of Randomness.

These tests are used to check random property that a random sequence is likely to have. Examples of useful statistical tests are five

basic tests, and they are: Frequency test, Serial test, Poker test, Runs test and Autocorrelation test [12]. The output of tests must be compared with passes values that illustrated in Table (6) to decide if the outputs of randomness tests are good for the sequences to pass.

In our algorithm, randomness tests are applied on different key sizes: 32, 128, and 256 bits. Tables (7), Table (8) and Table (8) are showing the results of applying randomness test on random image (Test image_ 1). Tables (10), Table (11) and Table (12) are showing the results of applying randomness test on random image (Test image_ 2).

From the results listed in the randomness tests tables, it is noticed that whenever the initialization block size (IV) and key size increase, almost randomness tests are passed, but when initialization block size (IV) and key size is few, many of test results are not passed (painted test with gray mean not passed). So initialization blocks size (IV) with a large size (64-bits or 128-bits) is recommended for generating random numbers and clipping key, while initialization block with small size (32- bits) is rejected .

7. Future Works

1. The initialization vector for hash function can be generated from any media (image, audio or even video), instead of string of characters.
2. Hash function could be applied to minimize the size of random key (key digest). This can be done by any widely used Hash functions (such as SHA family).
3. Generate 3_Dimension image randomly, and the key can be clipped from any section of it.
4. Random Art can be used to make this random image pleasing for Human Visual System.

8. Conclusion

Due to fundamental role of random key in the design of cryptography algorithms, a new method is proposed for generating random key. This key is clipped from random image (in any size and from any position of image). Hash technique is applied on input image and initialization vector, this vector is constructed by using random generator. Random image can be used to improve the accuracy and efficiency of key validation in public key infrastructures; also it can be used to improve user authentication systems when this random image can be replaced with texts string. Validation of random key is acceptable according to the results distortion measures and statistical tests of randomness.

References

[1] Alfred J. M., Paul C. O. and Scott A. V. "Handbook of Applied Cryptography ", fifth Addition, 2001.

[2] William Stallings, "Cryptography and Network Security Principles and Practices", third addition, 2003, Pearson Education, Inc.

[3] Hossein Bidgoli, John Wiley, "Key (cryptography)", 2004. Wikipedia. URL: http://en.wikipedia.org/wiki/Key_(cryptography)

[4] Jonathan B. Knudsen,"Java Cryptography", First Edition , O'Reilly Media, Inc., ISBN: 1-56582-402-8, 1888.

[5] Gary C. Kessler, "An Overview of Cryptography", (18 November 2008). URL: http://www.garykessler.net/library/crypto.html.

[6] Rafael Pass Scribe," Introduction to Cryptography", August 24, **2006.URL: http://www.cs.cornell.edu/courses/cs687/2006fa/lectures/lecture1.pdf**.

[7I lya Mironov," **Hash functions: Theory, attacks, and applications**"Microsoft esearch, Silicon Valley Campus, November 14, 2005.

[8] Damgard I.," **A design principle for hash functions**", Springer-Verlag, 1990.

[9] Neal Krawetz **,"Introduction to Network Security"**, Published by Charles River Media, an imprint of Thomson Learning, Inc.,2006

[10]Fabian Monrose, Michael K. Reiter, Qi Li, Susanne Wetzel**," Using Voice to Generate Cryptographic Keys"**, Murray Hill, NJ,USA.URL:

[11] Mauro Barni ,"**Document and image compression**", A CRC Press Books, 2006.

[12] Beker, H. J. and Piper, F. C., **"Secure Speech Commmunications",** Academic Press, New York, 1885.

Table (1) Random image is constructed from Test Image_1, using HASH Technique.

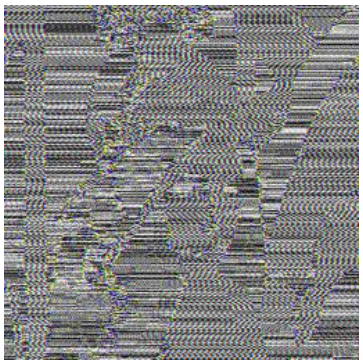
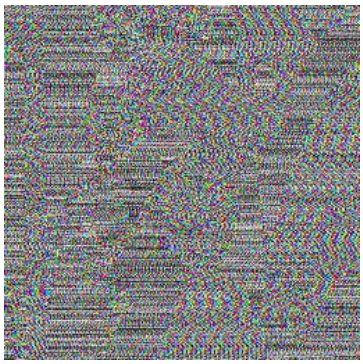
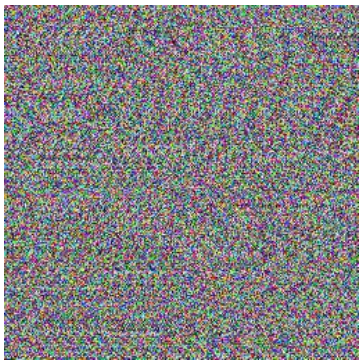
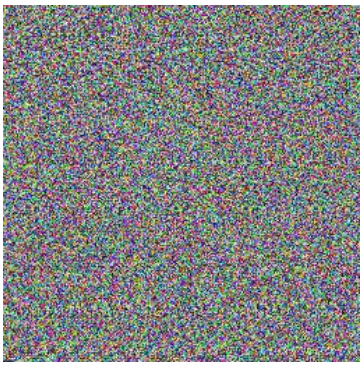
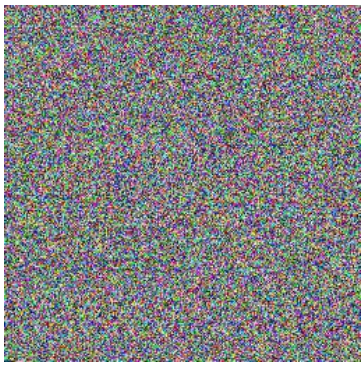
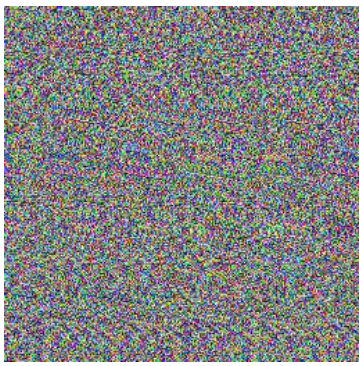
IV_1 = 1 Byte	*IV_2 = 5 Bytes*	*IV_3 = 25 Bytes*
		
IV_4 = 50 Bytes	*IV_5 = 100 Bytes*	*IV_6 = 150 Bytes*
		

Table (2) Random image is constructed from Test Image_2, using HASH Technique.

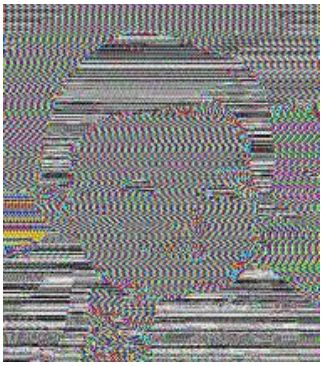
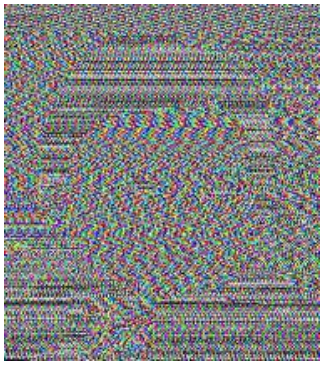
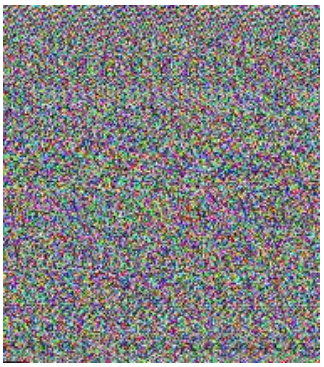
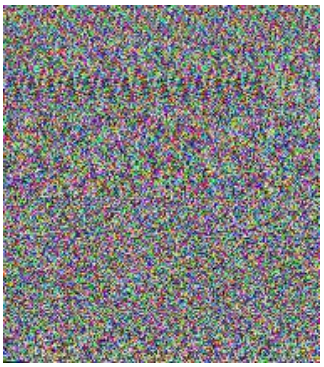
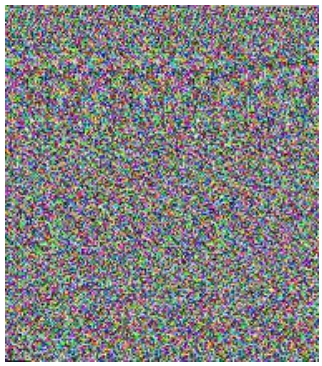
IV_1 = 1 Byte	*IV_2 = 5 Bytes*	*IV_3 = 25 Bytes*
		
IV_4 = 50 Bytes	*IV_5 = 100 Bytes*	*IV_6 = 150 Bytes*
		

Table (3) Distortion Measures [11].

Mean Square Error	$MSE = \frac{1}{H*W}\sum_{y=0}^{H-1}\sum_{x=0}^{W-1}\left(f(x,y)-f'(x,y)\right)^2$
Signal to Noise Ratio	$SNR = \frac{\sum_{y=0}^{H-1}\sum_{x=0}^{W-1}(f(x,y))^2}{\sum_{y=0}^{H-1}\sum_{x=0}^{W-1}(f(x,y)-f'(x,y))^2}$
Peak Signal to Noise Ratio	$PSNR = 10\log_{10}\left(\frac{(255)^2}{MSE}\right)$
Correlation Measure	$Similarity(A_{ij},B_{ij}) = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}a_{ij}b_{ij}}{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}a_{ij}^2}\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}b_{ij}^2}}$

Table (4) Distortion Measures of random images (Test image_ 1).

Test image _1	*Size (Byte)*	*MSE*	*PSNR*	*SNR*	*Correlation Measure*
IV_1	**1**	**7744.132**	**8.241**	**3.488**	**0.788**
IV-2	**5**	**7683.281**	**8.275**	**3.523**	**0.780**
IV_3	**25**	**7768.786**	**8.227**	**3.475**	**0.787**
IV_4	**50**	**768.550**	**8.227**	**3.474**	**0.788**
IV_5	**100**	**7777.648**	**8.222**	**3.470**	**0.788**
IV_6	**150**	**7788.318**	**8.216**	**3.463**	**0.788**

Table (5) Distortion Measures of random images (Test image_ 2).

Test image _2	*Size (Byte)*	*MSE*	*PSNR*	*SNR*	*Correlation Measure*
IV_1	**1**	**10003.823**	**8.128**	**2.77**	**0.854**
IV-2	**5**	**8583.833**	**8.3108**	**2.856**	**0.864**
IV_3	**25**	**8681.858**	**8.271**	**2.816**	**0.860**
IV_4	**50**	**8705.685**	**8.260**	**2.8058**	**0.858**
IV_5	**100**	**8618.322**	**8.300**	**2.845**	**0.858**
IV_6	**150**	**8677.308**	**8.273**	**2.818**	**0.862**

Table (6) Randomness Tests and their Passing Values [12].

Randomness Tests	*Passing Value*
Frequency Test	≤ 3 .84
Serial test	≤ 5 .99
Poker test	≤ 11 .1
Runs test	≤ 22 .362
Autocorrelation test	≤ 3 .48

Table (7) Results of Randomness tests on Test image_1 (Random Key Size =32 bits).

Initialization Vector (IV)	*Frequency Test*	*Serial Test*	*Poker Test*	*Run Test*		*Autocorrelation Test*									
				To	*T1*	*Shift-1*	*Shift-2*	*Shift-3*	*Shift-4*	*Shift-5*	*Shift-6*	*Shift-7*	*Shift-8*	*Shift-8*	*Shift-10*
IV-1	0.125	1.778	2	4.242	4.242	0.146	0.222	0.307	0.411	0.381	0.483	1.755	1.651	1.816	1.585
IV-2	6.125	8.681	18.066	1.75	4.875	0.3761	0.334	0.246	0.103	0.570	2.548	4.342	5.404	4.675	3.681
IV-3	0.125	2.8E-02	4.133	8.218	3.234	8.1E -03	0.880	0.062	1.318	5.5E -03	3.205	1.755	3.530	3.376	6.805
IV-4	0.125	1.004	1.467	2.688	5.438	0.1788	3.3E-02	0.0618	0.502	1.452	0.681	2.002	2.332	4.384	4.448
IV-5	2	3.774	3.6	3.00	5.88	1.277	5.8E-02	5.3E-02	1.877	2.118	0.114	0.128	0.184	5.561	0.315
IV-6	0.5	1.403	16.833	10.187	4.218	0.153	0.752	2.083	2.8E-02	0.444	0.280	1.283	1.188	0.605	1.0E-02

Table (8) Results of Randomness tests on Test image_1 (Random Key Size =64 bits).

Initialization Vector (IV)	*Frequency Test*	*Serial Test*	*Poker Test*	*Run Test*		*Autocorrelation Test*									
				To	*T1*	*Shift-1*	*Shift-2*	*Shift-3*	*Shift-4*	*Shift-5*	*Shift-6*	*Shift-7*	*Shift-8*	*Shift-8*	*Shift-10*
IV-1	0.0625	0.112	0.800	6.888	15.374	0.101	1.4E-02	3.3E-02	0.106	7.2E-02	5.1E-03	0.451	1.004	1.053	1.070
IV-2	1.5625	2.040	7.733	1.888	127.874	4.3E-02	8.7E-02	4.77E-03	0.235	7.1E-03	0.423	2.173	0.650	1.314	1.5
IV-3	0.0625	0.874	2.833	6.375	5.625	0.183	0.403	0.188	6.1E-02	8.7E-02	0.388	0.018	1.067	0.258	2.177
IV-4	0.25	0.178	2.4	3.625	4.000	8.3E-02	0.403	0.201	0	0.720	0.832	0.828	1.763	1.146	0.687
IV-5	0.25	0.306	4	4.250	8.250	0.087	0.827	0.423	0.444	0.888	0.570	0.365	1.058	2.138	1.313
IV-6	0.25	0.687	4.533	8.375	2.750	0.087	4.1E-02	0.847	0.182	0.148	4.4E-02	0.211	0.062	8.3E-02	0.133

*Computer Science Department,University of Technology/Baghdad,Iraq

Table (9) Results of Randomness tests on Test image_1 (Random Key Size =128 bits).

Initialization Vector (IV)	*Frequency Test*	*Serial Test*	*Poker Test*	*Run Test*		*Autocorrelation Test*									
				To	*T1*	*Shift-1*	*Shift-2*	*Shift-3*	*Shift-4*	*Shift-5*	*Shift-6*	*Shift-7*	*Shift-8*	*Shift-8*	*Shift-10*
IV-1	0.5	0.808	3.16	11.124	8.874	3.5E-02	1.8E -03	0.017	1.414-02	0.256	1.8E -02	0.637	3.2E -02	0.418	0.761
IV-2	1.531	1.767	14.68	5.562	84.062	2.1E 02	0.485	5.1-02	1.347	1.857-3	0.368	1.002	0.136	0.808	1.020
IV-3	0.125	0.528	10.84	8.625	8.062	0.108	0.400	0.081	0.120	0.336	0.180	8.8E-2	0.515	5.0E -03	0.364
IV-4	0.031	1.882	1.624	8.375	4.812	0.583	0.430	0.378	0.328	0.158	0.668	0.013	1.665	0.144	1.008
IV-5	0.781	2.077	11.882	2.625	17.187	0.260	0.167	8.4E -02	1.068	0.158	0.1761	0.103	0.517	0.583	0.418
IV-6	0.781	1.510	3.672	7.187	2.562	0.142	7.6E-06	0.180	4.8E -04	0.382	0.241	0.108	0.327	0.156	0.702

Table (10) Results of Randomness tests on Test image_2 (Random Key Size =32 bits).

Initialization Vector (IV)	*Frequency Test*	*Serial Test*	*Poker Test*	*Run Test*		*Autocorrelation Test*									
				To	*T1*	*Shift-1*	*Shift-2*	*Shift-3*	*Shift-4*	*Shift-5*	*Shift-6*	*Shift-7*	*Shift-8*	*Shift-8*	*Shift-10*
IV-1	1.125	3.617	5.733	10.625	2.1875	0.617	0.352	1.058	0.145	5.0E-02	8.3E-02	0.128	3.010	5.561	2.834
IV-2	0.125	0.746	2	2.234	1.882	7.2E-02	3.3E-02	2.1E-02	3.7E-03	2.841	0.210	5.683	1.050	1.280	4.832
IV-3	0	8.677	4.133	7.468	4.734	5.2E-03	0.384	1.081	0.502	0.264	0.684	3.453	0.782	4.3841	2.216
IV-4	1.125	3.875	6.8	2.375	1.875	1.600	0.587	8.8E-03	0.187	0.705	8.3E-02	0.885	1.170	1.370	0.338
IV-5	0.5	0.887	2	3.484	7.234	6.6E-02	0.278	8.1E-02	1.286	8.2E-03	1.881	2.160	2.331	7.021	4.448
IV-6	2	4.806	5.733	3.125	1.838	0.488	0.587	8.8E-03	2.5E-02	0.350	1.625	1.562	0.167	0.220	0.102

Table (11) Results of Randomness tests on Test image_2 (Random Key Size =64 bits).

Initialization Vector (IV)	Frequency Test	Serial Test	Poker Test	Run Test		Autocorrelation Test									
				To	T1	Shift-1	Shift-2	Shift-3	Shift-4	Shift-5	Shift-6	Shift-7	Shift-8	Shift-8	Shift-10
IV-1	3.0625	4.000	5.067	11.123	1.124	0.517	0.338	3.8E-02	0.245	1.1E-02	0.304	1.7E-02	0.858	4.355	0.822
IV-2	0.0625	0.238	1.6	4.374	3.374	1.7E-03	2.8E-02	0.864	0.437	1.247	0.757	1.465	0.823	1.053	3.022
IV-3	0.25	1.067	1.6	6.625	3.750	0.245	2.8E-02	1.8E-02	3.7E-02	0.466	7.3E-02	2.324	0.107	0.500	0.133
IV-4	3.0625	5.874	4.533	7.888	5.121	1.183	3.1E-02	0.236	0.250	0.250	0.672	1.582	0.288	3.8E-02	0.411
IV-5	2.25	1.888	5.6	8.248	4.248	5.3E-02	7.8E-02	2.5E-02	0.236	0.104	1.310	1.550	1.135	4.85	1.5
IV-6	2.25	5.036	4.8	3.873	1.748	0.583	8.1E-03	2.1E-02	0.038	0.350	8.7E-02	0.128	1.058	1.5E-03	0.136

Table (12) Results of Randomness tests on Test image_2 (Random Key Size =128 bits).

Initialization Vector (IV)	Frequency Test	Serial Test	Poker Test	Run Test		Autocorrelation Test									
				To	T1	Shift-1	Shift-2	Shift-3	Shift-4	Shift-5	Shift-6	Shift-7	Shift-8	Shift-8	Shift-10
IV-1	2.531	2.784	3.672	8.812	2.562	1.2E-02	0.262	0.341	0.427	3.8E-02	0.208	0.147	0.571	1.536	0.118
IV-2	0.125	0.402	1.624	3.312	8.875	0.154	7.8E-03	0.05	0.806	1.282	0.668	0.464	0.531	0.828	1.832
IV-3	4.5	5.350	6.488	18.874	4.062	6.6E-02	2.6E-03	0.060	0.807	1.258	2.740	4.1E-02	0.450	1.2E-02	0.446
IV-4	2.531	8.587	8.844	58.000	7.687	2.432	0.287	1.5E-02	3.2E-03	1.657	1.235	4.1E-02	0.450	1.2E-02	0.446
IV-5	1.531	1.578	4.312	2.062	2.488	1.3E-03	0.345	0.684	0.767	0.022	0.846	4.1E-02	0.450	1.2E-02	0.446
IV-6	0.031	0.118	1.112	12.624	1.8124	1.3E-03	6.6E-02	1.3E-02	0.278	0.152	1.426	2.537	0.286	0.283	0.327

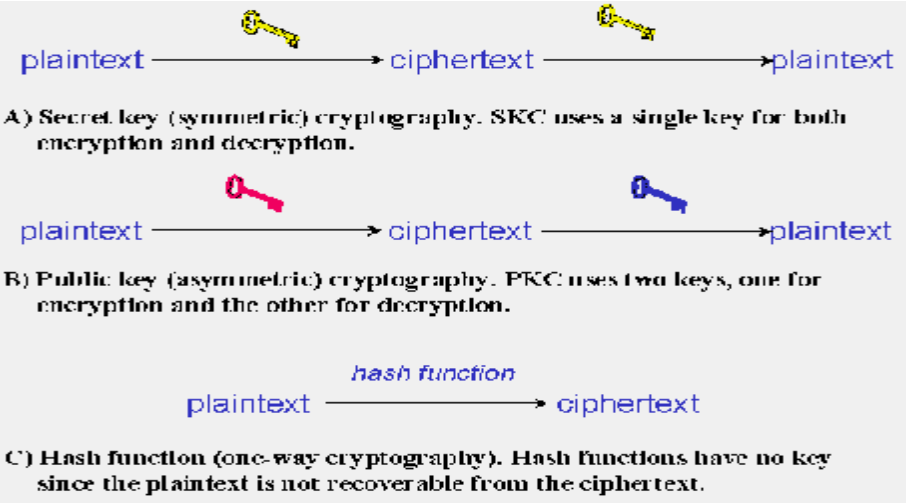


Figure (1) Three types of cryptography: A) secret-key, B) public key and C) hash function [5].

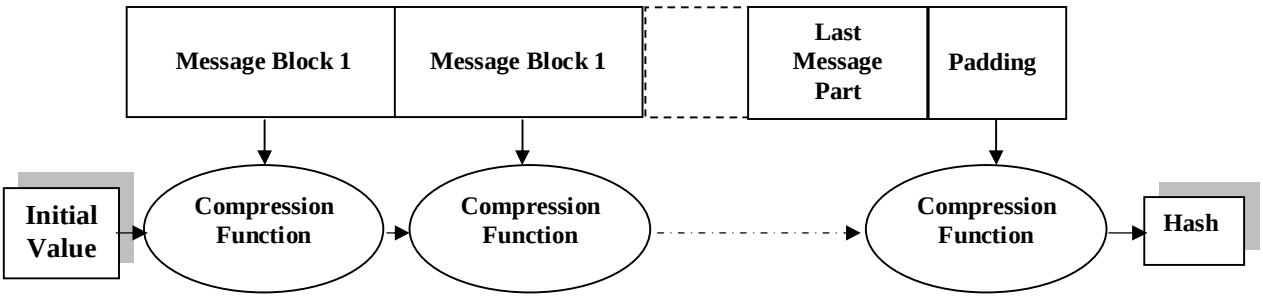


Figure (2) Damgard /Merkle iterative structure for hash functions [8].

Algorithm (1) : Padding Password

Input : Password.
Output : Padded password with required length

Step 1 : Enter the required length of password (R_len)
Step 2 : K=0
Step 3: While Length of (Password) < (R_len)
k= k+ Length of (Password)
m = k * 257 + 313
Password = Password + Chr(32 + (m Mod 86))
End While.

Algorithm (2): Generation of Three Parameters(R, N, S).

Input : BITS (16) is string represent the bases of binary numbers.
Output : Random parameters: R (0...65535), N (0...31), and S (0...127).

Step 1 : Convert padded password to binary code, since the desired length is 25, the output is a stream of bits B with length 175 B (175).
Step 2: Choose arbitrary bits from b(175) to form initial value K:
K= B (10) + 2 × B (61) + 4 × B (88) + 8 × B (175)
Step 3: For J = 0 to 14
If B (K) = 1 Then R = R + Bit (J)
K = K + 18
If K > 174 Then K = K - 174
End For J

Algorithm (3): Construction Sequence Generator

Input : Random Parameters (R,N,S) , length of sequence generator.
Output : Five Sequence Generator SQ1(L1), SQ2(L2), SQ3(L3), SQ4(L4), SQ5(L5).

Step 1 : K1 = S
Step 2: L1=252
Step 3: Randomize R.
Step 3: For J = 0 to L1
For K = 0 to N
X = Rnd
End of K
SQ1 (J) = 255 × X
End of J

Algorithm (4) : Random Image generator.
Input : Input Image , Initialization Vector .
Output : Random Image .

Step 1 : Divide input image into blocks , each block with 10 bytes size

Step 2: generate random number with size = 10 bytes , it's considered as IV
Step 3: For each block of input image (Bi) do
Generated _Block = Bi XOR IV
Store Generated _Block in Random image
IV= Generated _Block
Next For

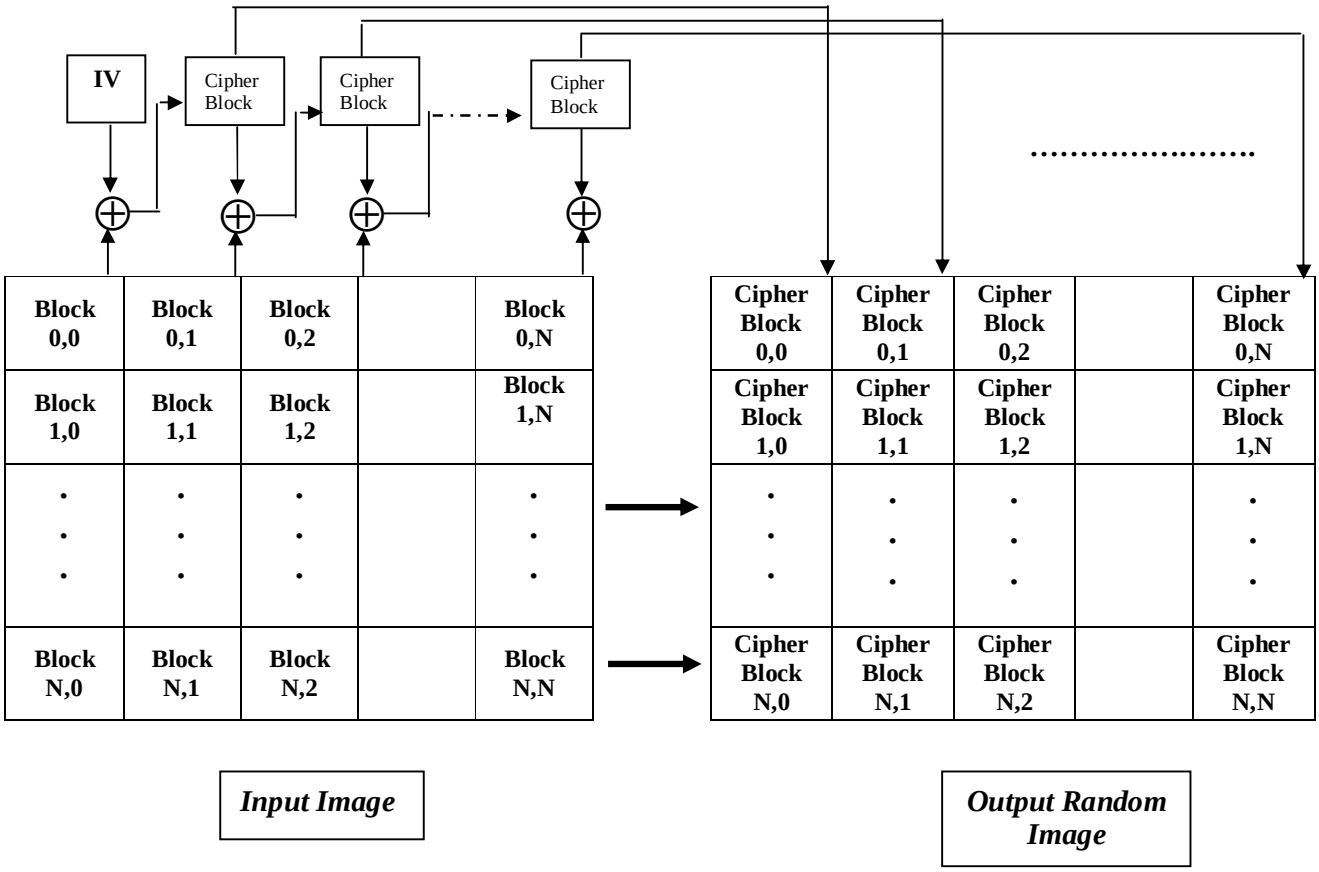


Figure (3) Hash Technique (IV is XOR-ed with input image to construct random image).

Figurer (4) : Test image_1 (Gray image)

Figurer (5): Test image _2 (Color image).