A

MAJOR PROJECT REPORT

ON

PSEUDORANDOM BINARY SEQUENCE GENERATION FOR STREAM CIPHERS

Submitted in the partial fulfillment of the requirements
for the award of the degree of

MASTER OF TECHNOLOGY
(INFORMATION SYSTEM)

Submitted By:
VIJETA RANI
(17/IS/09)

Under the Guidance of
N. S. RAGHAVA
Associate Professor
Dept. of Information Technology

DELHI TECHNOLOGICAL UNIVERSITY
(DEPARTMENT OF INFORMATION TECHNOLOGY)
BAWANA ROAD, DELHI-110042
SESSION: 2009-2011

CERTIFICATE

This is to certify that **Ms. Vijeta Rani (17/IS/09)** has carried out the major project titled "**Pseudorandom Binary Key Generation for Stream Ciphers**" as a partial requirement for the award of Master of Technology degree in Information Systems by Delhi Technological University.

The major project is an authentic piece of work carried out and completed under my supervision and guidance during the academic session **2009-2011** at Delhi Technological University (Formerly Delhi College of Engineering). The matter contained in this report has not been submitted elsewhere for the award of any other degree.

N. S. Raghava
(Project Guide)
Associate Professor
Department of Information Technology
Delhi Technological University
Shahbad Daulatpur, Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

I express my gratitude to my major project guide, **N. S. Raghava**, Associate Professor, for the valuable support and guidance he provided in making this major project. It is my pleasure to record my sincere thanks to my respected guide for his constructive criticism and insight without which the project would not have shaped as it has.

I humbly extend my words of gratitude to **Prof. O. P. Verma**, head of the department, and other faculty members of IT department for providing their valuable help and time whenever it was required.

Vijeta Rani
Roll No: 17/IS/09
M. Tech (Information Systems)
Email: vijeta3feb@gmail.com

ABSTRACT

Pseudorandom binary sequences find their application in diverse fields but security and cryptography is probably the best known field of their application. One-Time Pad (OTP) is a simple, fast and the most secure encryption algorithm. It provides the perfect secrecy. The encryption-decryption process of the OTP is based on exclusive-or function computed on the plaintext/ciphertext and the key bits. The requirements for the OTP key are that: it must be a cryptographically strong truly random or pseudorandom binary sequence; must be as long as plaintext size; and must not be reused. The difference between a truly random and a pseudorandom sequence is that the truly random sequence is generated with the help of non-deterministic physical phenomenon but the pseudorandom sequence is generated from some deterministic mechanism and a seed value. In case of pseudorandom binary sequences, given the same seed the pseudorandom number generator will always output the same sequence of numbers or bits. The fundamental difficulty with a truly random sequence is its generation and distribution. Therefore pseudorandom sequences are a popular choice for the practical implementation of the OTP scheme.

Many researchers have devoted their time and effort to the family of shift register based pseudorandom sequence generators. But they could not gain a key sequence having very large period equal to the plaintext length. They also tried the complex versions of shift registers but it is yet not very useful and secure to generate a sequence large enough to encrypt an audio or video

file. Moreover, these sequences do not satisfy the statistical properties of random numbers to that great extent.

To find a better alternative to the shift registers and to generate a very long cryptographically strong pseudorandom binary key at a very low cost is the main objective behind this research work. A few algorithms are proposed in this report to generate a long cryptographically strong pseudorandom binary key for stream ciphers using the multimedia files available on the Internet. Both the authorized sender and receiver download a file from the Internet whose link is shared between them through a secure medium. This file can be in the form of text, audio, video or image and contains huge amount of redundancy. They use this file as the seed or the input to the proposed algorithms and generate a cryptographically strong pseudorandom binary key file from it. The key files obtained from the algorithms' implementation are statistically validated for the practical implementation of OTP using the well known NIST and ENT test suites.

Table of Contents

CHAPTER 1

PSEUDORANDOM BINARY KEY GENERATION

1.1 Introduction

A strong cryptographic algorithm is a basic requirement for any cryptosystem. A cryptographic algorithm can be considered strong only if either it is unconditionally secure or it is computationally secure. A cryptographic algorithm is said to be *unconditionally secure* if the information in the cipher text cannot help in determining the plaintext uniquely [4]. It is said to be *computationally secure* if the cost of breaking the cipher exceeds the value of encrypted information or the time required to break the cipher exceeds the useful lifetime of the information [4]. Only One-time Pad, which is a stream cipher algorithm, is unconditionally secure (or provides perfect secrecy) [3].

In stream cipher algorithms, the plain text bits are XORed with the key bits to produce cipher text bits, which are again XORed with key bits at the receiver side to recover the plain text bits. In One-time Pad, a truly random key is used for only one time whose length is equal to the plaintext. The practical difficulty of the One-time Pad is that the key, which must be randomly generated and communicated over a secure channel, must be as long as the plaintext in order to ensure perfect secrecy [7]. In other words, the cost of key generation and distribution cannot be ignored. Perfect secrecy is defined in [5].

There are two ways of generating random bits:

1. Generating truly random bits using physical mechanisms where the whole bit sequence is securely transmitted.
2. Generating pseudorandom bits using some seed where only the seed is securely transmitted.

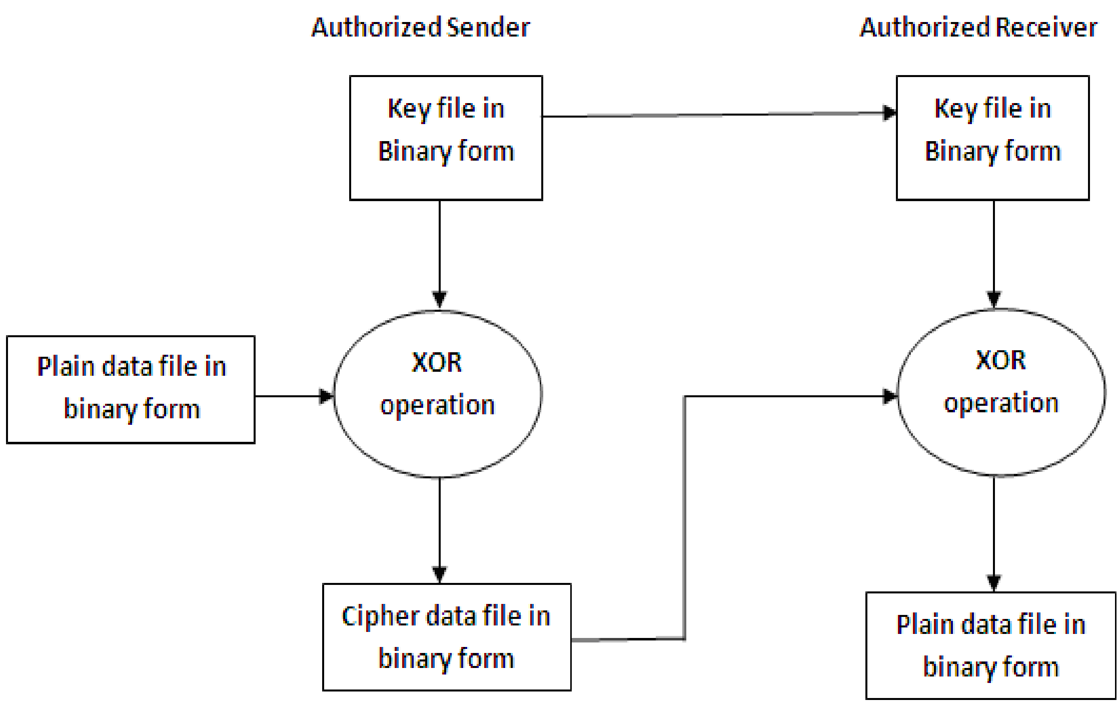

Fig 1.1 Encryption-Decryption Processes of Stream Cipher Algorithms

The generation of truly random bits is difficult, time consuming and expensive and the distribution of whole of these bits through a secure channel is not practical especially when the size of the bit sequence is as large as the plaintext size. The problem identified above can be avoided if a pseudo-random number (or bit) generator is used instead of truly random number (or bit) generator.

A pseudorandom number generator (PRNG), also known as deterministic random bit generator (DRBG), is an algorithm for generating a sequence of numbers that approximates the properties of random numbers [38]. The sequence is not truly random in that it is completely determined by a relatively small set of initial values, called the PRNG's state [38]. A PRNG can be started from an arbitrary starting state using a seed state. It will always produce the same sequence thereafter when initialized with that state [38]. Careful mathematical analysis is required to have any confidence a PRNG generates numbers that are sufficiently "random" to suit the intended use [38].

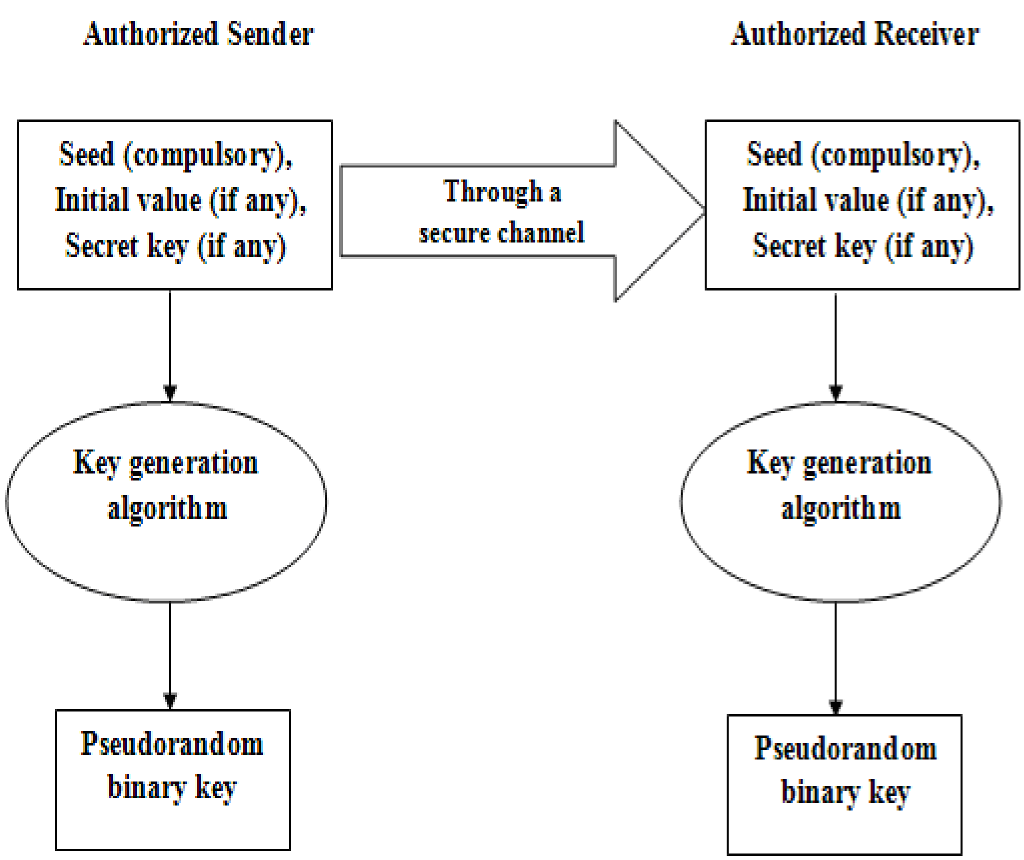

Fig 1.2 Pseudorandom Binary Key Generation Process

1.2 Overview of Existing Methods

A system named, **Vernam Cipher** [1, 2], was introduced by an AT&T engineer named Gilbert Vernam in 1918. His system works on binary data rather than letters. The ciphertext is generated by performing the bitwise XOR of the plaintext and the key. Because of the properties of the XOR, decryption simply involves the same bitwise operation. The essence of this technique is the means of construction of the key [4].

An Army Signal Corp officer, Joseph Mauborgne, proposed an improvement to the Vernam cipher that yields the ultimate in security. Mauborgne suggested using a random key that is as long as the message, so that the key need not be repeated. In addition, the key is to be used to encrypt and decrypt a single message, and then is discarded. Each new message requires a new key of the same length as the new message. Such a scheme, known as a **One-time Pad**, is unbreakable [3]. It produces random output that bears no statistical relationship to the plaintext. Because the ciphertext contains no information whatsoever about the plaintext, there is simply no way to break the code [3, 4].

There are certain design considerations for stream ciphers [5], which are given below.

1. The encryption sequence should have a large period
2. The encryption sequence should be unpredictable from its previous state. To ensure this property, the sequence should have a large complexity and proper distribution of ones and zeros.
3. There should be large variability of the possible keys.

There are many techniques for generating stream cipher sequences. An overview of some of them is given in [6]. The text includes the description of real random and pseudorandom sequence generators.

Real Random-Sequence Generators include RAND Tables, Using Random Noise, Using the Computer's Clock, Measuring Keyboard Latency, Biases and Correlations, Distilling Randomness.

Pseudo-Random Sequence Generators include Linear Congruential Generators, Linear Feedback Shift Registers (LFSR), Feedback with Carry Shift Registers, Nonlinear-Feedback Shift Registers, RC4, SEAL, WAKE, PKZIP, A5, Hughes XPD/KPD, RSA, Shamir's Pseudo-Random-Number Generator, Blum-Micali Generator, Blum-Blum-Shub Generator, Nanoteq, Rambutan, Additive Generators (like Fish, Pike, Mush), Gifford, Algorithm M, Pless Generator, Cellular Automaton Generator, 1/p Generator, crypt(1), Rip van Winkle Cipher, Diffie's Randomized Stream Cipher, Maurer's Randomized Stream Cipher.

Stream ciphers that use LFSR are Geffe Generator, Generalized Geffe Generator, Jennings Generator, Beth-Piper Stop-and-Go Generator, Alternating Stop-and-Go Generator, Bilateral Stop-and-Go Generator, Threshold Generators, Self-Decimated Generators, Multispeed Inner-Product Generator, Summation Generator, DNRSG, Gollmann Cascade, Shrinking Generator, Self-Shrinking Generator.

Many researchers worked to find new techniques for pseudorandom sequence generation. Several encryption based pseudorandom sequences have also been proposed. A few of them are: generating pseudorandom sequence by encrypting the seed (possibly some memory attribute like instruction address) [22], Elgamal discrete logarithm pseudorandom sequence generation [20] and JCA based pseudorandom sequence generation [23]. Pseudorandom sequence generation from DNA information [24] is another growing field of interest. Other researchers have contributed some more techniques for pseudorandom sequence generation [28, 40, 44 – 52] and their applications [25, 26].

Most practical stream-cipher designs center around LFSRs. The problem with LFSRs is that they are very inefficient and slow in software [6]. Sparse feedback polynomials are avoided as they facilitate correlation attacks and dense feedback polynomials are inefficient [6]. Analyzing stream ciphers is often easier than analyzing block ciphers [6]. For example, one important metric used to analyze LFSR-based generators is linear complexity, or linear span. This is defined as the length, n, of the shortest LFSR that can mimic the generator output [5, 6]. Any sequence generated by a finite-state machine over a finite field has a finite linear complexity [6]. Linear complexity is important because a simple algorithm, called the Berlekamp-Massey algorithm, can generate this LFSR after examining only $2n$ bits of the keystream [5, 6]. Once this LFSR is generated, the stream cipher is broken. One or more of the internal output sequences, often just outputs of individual LFSRs, can be correlated with the combined keystream and attacked using linear algebra. Often this is called a correlation attack or a divide-and-conquer attack.

There are other general attacks against keystream generators. The linear consistency test attempts to identify some subset of the encryption key using matrix techniques. There is also the meet-in the-middle consistency attack [6]. The linear syndrome algorithm relies on being able to write a fragment of the output sequence as a linear equation. There is the best affine approximation attack and the derived sequence attack [6]. The techniques of differential cryptanalysis have even been applied to stream ciphers, as has linear cryptanalysis [6].

Some researchers have worked for generating pseudorandom sequences from microphone input [35]. They have also proposed algorithm for it. According to their algorithm, the eight bit of each byte is appended to the sequence and the high 7 bits of every byte are discarded. They have tested their algorithm using ENT test and the test results show that the sequences generated from algorithm are random enough. But according to our observation, this algorithm will not be applicable to files of any type as it will not generate highly random sequences for all files. For instance, if a file contains bytes which are actually random and different from each other, but the eight bit of each byte is zero or one for the whole file, then the sequence generated from such input will always be a sequence of all zeros or all ones. Such a sequence has zero randomness. Moreover the truncation of high bits generates a very small length sequence as compared to the input file.

CHAPTER 2

INVERSION-COMPRESSION METHOD

2.1 Introduction

This is the first proposed algorithm and is a compression based algorithm. The authorized sender and receiver can share their very large size data over an unsecure channel at a very low cost using this algorithm. For this, the authorized sender and the authorized receiver share a link on the Internet from where a file can be downloaded. This link must be shared using a very secure form of communication so that an unauthorized receiver is not able to see it. The authorized sender and receiver also decide the compression algorithm to be used during the process and share the secret value n between them through the same secure channel. Then they download the file from the Internet using this link, which is termed as the source file. Then both the authorized sender and receiver generate the key file using this source file. This key file is XORed with the plain data file to generate a cipher data file. Then this cipher data file is transmitted over the unsecure channel.

The information that goes from the authorized sender to the authorized receiver through the secure channel or medium includes: the file download link, the compression algorithm and the secret value n.

The information that goes from the authorized sender to the authorized receiver through the insecure channel or medium includes: the cipher data file encrypted using the key file generated from the proposed method.

2.2 Methodology

The key generation algorithm consists of the following steps.

Step 1: Input the source binary file in binary read mode. Open two other binary files, say X and Y in binary write mode.

Step 2: Segment the source file in one byte segments each.

Step 3: Remove those bytes from the source file whose ASCII value is either 0 or 255. Store the result in X.

Step 4: Close X.

Step 5: Open X in binary read mode.

Step 6: Remove all those bytes from X, which duplicate to their immediate predecessor bytes and store the resultant bytes in Y.

Step 7: Close X and Y.

Step 8: Open X in binary write mode and Y in binary read mode.

Step 9: Scan Y, bit by bit, to get a sequence of n consecutive zeros or n consecutive ones, where $3 \leq n \leq 14$. Note that there will be no sequence of 15 or more consecutive ones or zeros as the bytes with ASCII values 0 or 255 are already removed from the file. On getting such a sequence, invert the final bit of the sequence. As a result, there will be no sequence in the file, which is composed of more than n-1 consecutive zeros or n-1 consecutive ones. This n is kept secret

Step 10: Store the resultant bytes in X.

Step 11: Open X in binary read mode and Y in binary write mode.

Step 12: Compress X using some standard compression software like Win RAR or Win Zip. This increases randomness of the file. The standard compression algorithms like Win RAR or Win ZIP are preferred for the process as they provide optimum compression of the file. If any other compression technique provides better results i.e. adds more randomness to the file then that compression algorithm can be substituted at this step.

Step 13: Store the resultant bytes in Y. The resultant file Y serves as the key file during encryption.

Step 14: Close all files.

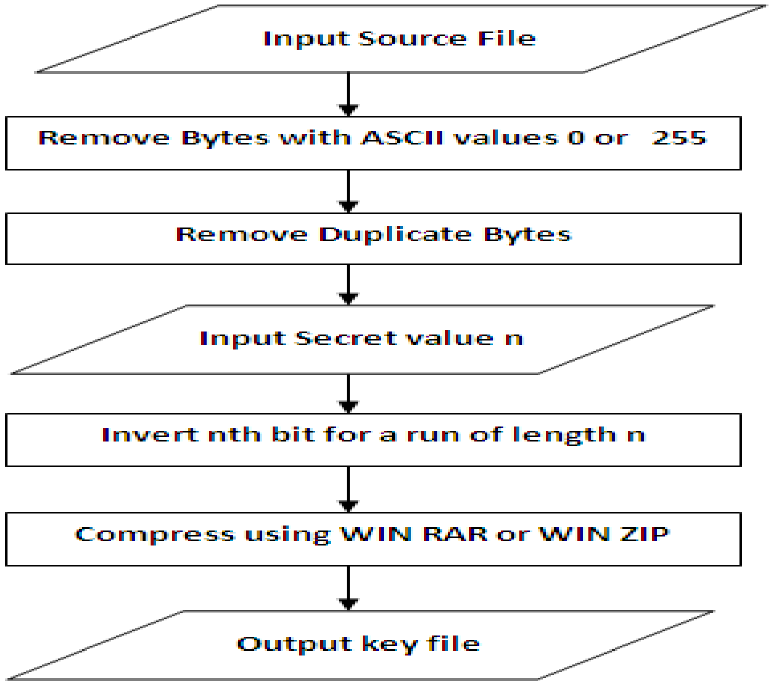

Fig. 2.1 Flowchart of the Inversion-Compression method

The significance of the major steps of the algorithm is given as follows:

Removal of bytes with ASCII values 0 or 255: In this step, the bytes with ASCII values 0 or 255 are removed from the source file. Bytes with ASCII values 0 or 255 facilitate the cryptanalyst to attack the ciphertext. The cryptanalyst will simply apply XOR operation on the ciphertext file and the file containing all zeros or ones. This will result into the plaintext file containing original characters at the position of bytes with ASCII values 0 or 255 in the key file. Thus applying this step is essential. Moreover, this step removes long runs of zeros and ones**.**

Removal of successive duplicate bytes: In this step, those bytes which are identical to their immediate predecessor are removed from the source file.

Duplicate bytes increase the redundancy of the source file and hence should be removed to make it more random. In a way, duplicate bytes reduce the period of the sub key. Thus removing duplicate bytes is essential. Only successive duplicate bytes are removed and not the duplicate bytes of the whole file are removed because on doing so the file size will reduce to a maximum of 256 bytes. This is because only 256 byte patterns are available. A file with a maximum size of 256 bytes is not at all random as the attacker will simply use 256*256 combinations to apply a brute force attack.

Inversion: In this step, the source file is scanned, bit by bit, to get a sequence of n consecutive zeros or n consecutive ones, where $3 \leq N \leq 14$. On getting such a sequence, the final bit of the sequence is inverted. As a result, there will be no sequence in the file, which is composed of more than N-1 consecutive zeros or N-1 consecutive ones. This N is kept secret. There will be no sequence of 15 or more consecutive ones or zeros as the bytes with ASCII values 0 or 255 are already removed from the file. Inversion for $N<3$ is not useful at all. Inversion for N=1 corresponds to the inversion of all the bits of the file. Inversion for N=2 corresponds to a file having alternate 1 and 0 bits, which has zero randomness.

Compression: In this step, the file is finally compressed using standard compression algorithms like Win RAR or Win ZIP. This removes the remaining redundancy from the file. The standard compression algorithms like Win RAR or Win ZIP are preferred for the process as they provide optimum compression of the file. If any other compression technique provides better results i.e. adds more randomness to the file then that compression algorithm can be substituted at this step.

2.3 Results

The proposed algorithm was implemented in C programming language and used WIN RAR [37] compression software to generate the pseudorandom binary key files from various types of source files (i.e. from the text, audio, video and image files). The change in the size of the files after each step is given in table 2.1.

We also performed ENT [31] and NIST [32] statistical tests on the source files and the corresponding output files of the key generation algorithm to find the amount of randomness in the files. The ENT and the NIST tests are performed using the standard testing software available online on the official websites of these tests.

The results of the ENT tests are given in the tables 2.2, 2.3, 2.4 and 2.5 for text, image, audio and video files respectively. The results of NIST tests for the text file are given in tables 2.6, 2.7, 2.8 and 2.9.

Table 2.1 Size of the files after each step

File Type	Size of SF	Size of NRF	Size of DRF	Size of KF (Varied because of compression)
Text	11.8 KB	11.8 KB	11.6 KB	4.48 KB
Audio	94.7 KB	61.3 KB	61 KB	55 KB
Video	3.97 MB	3.50 MB	3.47 MB	3.41 MB
Image	63.7 KB	61.5 KB	58 KB	44.7 KB

2.3.1 ENT Tests

Table 2.2 Results of ENT tests on Text File

TEXT FILE							
File type	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
Totally Random	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
UCF	4.364403	45	0.01	93.5753	4.000000000	27.32	0.005331
CF	7.953960	0	3.10	129.2470	3.125964010	0.50	0.007876
Null Bytes Removed File							
UCF	4.364403	45	0.01	93.5753	4.000000000	27.32	0.005331
CF	7.954017	0	3.06	129.3612	3.100257069	1.32	0.007560
Duplicate Bytes Removed File							
UCF	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
CF	7.958318	0	25.71	128.2676	3.152941176	0.36	0.022398
File after applying inversion to a sequence of length two							
UCF	0.000000	100	0.01	85.000	4.000000000	27.32	UNDEF
CF	4.948120	38	0.01	78.3125	3.750000000	19.37	0.442797
File after applying inversion to a sequence of length three							
UCF	4.021938	49	0.01	136.0150	3.379032258	7.56	-0.207247
CF	7.957984	0	65.79	124.4317	3.170243205	0.91	0.000763
File after applying inversion to a sequence of length four							
UCF	4.313830	46	0.01	101.0115	3.961693548	26.10	-0.064890
CF	7.960601	0	44.16	127.2657	3.093750000	1.52	0.049017
File after applying inversion to a sequence of length five							
UCF	4.511224	43	0.01	97.3783	3.870967742	23.22	0.030595
CF	7.956798	0	9.78	129.1454	3.006435006	4.30	0.048245
File after applying inversion to a sequence of length six							
UCF	4.882123	38	0.01	113.4824	3.397177419	8.14	-0.410593
CF	7.959954	0	25.25	125.8664	3.225000000	2.65	-0.006100
File after applying inversion to a sequence of length seven							
UCF	4.364658	45	0.01	93.9212	4.000000000	27.32	-0.045451
CF	7.952078	0	2.30	127.3687	3.047120419	3.01	0.005032
File after applying inversion to a sequence of length eight							
UCF	4.372235	45	0.01	93.8406	4.000000000	27.32	-0.035130
CF	7.957496	0	21.48	127.4739	3.142483660	0.03	0.002176
File after applying inversion to a sequence of length nine							
UCF	4.372949	45	0.01	93.8379	4.000000000	27.32	-0.034668
CF	7.956418	0	19.86	125.8027	3.101827676	1.27	0.005942
File after applying inversion to a sequence of length ten							
UCF	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
CF	7.957868	0	21.34	128.2182	3.064052288	2.47	0.022256
File after applying inversion to a sequence of length eleven							
UCF	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
CF	7.958183	0	23.32	128.1979	3.064052288	2.47	0.023221
File after applying inversion to a sequence of length twelve							
UCF	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
CF	7.958104	0	22.88	128.1511	3.058823529	2.63	0.022072
File after applying inversion to a sequence of length thirteen							

UCF	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
CF	7.958293	0	24.35	128.1898	3.064052288	2.47	0.022553
File after applying inversion to a sequence of length fourteen							
UCF	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
CF	7.958303	0	24.79	128.1683	3.064052288	2.47	0.022492

Table 2.3 Results of ENT tests on Image File

IMAGE FILE							
File type	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
Totally Random	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
UCF	7.184481	10	0.01	126.8271	3.504828474	11.56	0.435878
CF	7.994161	0	0.01	128.9419	3.096559114	1.43	0.049624
Null Bytes Removed File							
UCF	7.190326	10	0.01	124.0478	3.596536302	14.48	0.573973
CF	7.993797	0	0.01	129.7162	3.092432832	1.56	0.043240
Duplicate Bytes Removed File							
UCF	7.213525	9	0.01	123.9359	3.612877182	15.00	0.548074
CF	7.993735	0	0.01	126.3317	3.164069661	0.72	0.044904
File after applying inversion to a sequence of length two							
UCF	0.000000	100	0.01	85.0000	4.000000000	27.32	UNDEF
CF	4.683108	41	0.01	64.2540	3.809523810	21.26	0.327583
File after applying inversion to a sequence of length three							
UCF	5.660456	29	0.01	126.9927	3.795741245	20.82	0.100742
CF	7.994063	0	3.38	126.8655	3.147127200	0.18	0.005054
File after applying inversion to a sequence of length four							
UCF	6.770797	15	0.01	125.7696	3.649207791	16.16	0.381586
CF	7.994082	0	0.01	126.9810	3.147245315	0.18	0.033214
File after applying inversion to a sequence of length five							
UCF	7.133488	10	0.01	125.4563	3.603189020	14.69	0.459713
CF	7.994796	0	0.17	127.9209	3.149455800	0.25	0.040455
File after applying inversion to a sequence of length six							
UCF	7.251777	9	0.01	123.8837	3.612069836	14.98	0.506082
CF	7.994911	0	0.21	128.4259	3.117962116	0.75	0.035692
File after applying inversion to a sequence of length seven							
UCF	7.250141	9	0.01	124.3192	3.624987385	15.39	0.545376
CF	7.995300	0	3.17	126.9261	3.180464873	1.24	0.039492
File after applying inversion to a sequence of length eight							
UCF	7.270602	9	0.01	124.0301	3.594711878	14.42	0.538111
CF	7.994054	0	0.01	126.2493	3.182504556	1.30	0.035721
File after applying inversion to a sequence of length nine							
UCF	7.233893	9	0.01	123.9347	3.614088203	15.04	0.550350
CF	7.994059	0	0.01	126.3912	3.174271147	1.04	0.041389
File after applying inversion to a sequence of length ten							
UCF	7.225757	9	0.01	123.9300	3.613684529	15.03	0.549011
CF	7.994625	0	0.02	126.6920	3.148090005	0.21	0.039094
File after applying inversion to a sequence of length eleven							

UCF	7.219178	9	0.01	123.9324	3.613684529	15.03	0.548359
CF	7.994027	0	0.01	126.3366	3.144802304	0.10	0.040151
File after applying inversion to a sequence of length twelve							
UCF	7.214809	9	0.01	123.9353	3.612877182	15.00	0.548112
CF	7.994066	0	0.01	126.2537	3.134965310	0.21	0.044771
File after applying inversion to a sequence of length thirteen							
UCF	7.213700	9	0.01	123.9358	3.612877182	15.00	0.548083
CF	7.994056	0	0.01	126.3751	3.161560618	0.64	0.042576
File after applying inversion to a sequence of length fourteen							
UCF	7.213525	9	0.01	123.9359	3.612877182	15.00	0.548074
CF	7.993722	0	0.01	126.3322	3.147832919	0.20	0.044891

Table 2.4 Results of ENT tests on Audio File

AUDIO FILE							
File type	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
Totally Random	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
UCF	6.200233	22	0.01	125.2086	2.295751129	26.92	0.174946
CF	7.944631	0	0.01	120.9038	3.231529657	2.86	0.122647
Null Bytes Removed File							
UCF	7.580974	5	0.01	125.6436	2.861509074	8.92	0.026084
CF	7.994756	0	0.01	129.3176	3.102113066	1.26	0.016966
Duplicate Bytes Removed File							
UCF	7.585443	5	0.01	125.6610	2.866961029	8.74	0.019331
CF	7.995311	0	0.01	129.3084	3.061501423	2.55	0.000428
File after applying inversion to a sequence of length two							
UCF	0.000000	100	0.01	85.0000	4.000000000	27.32	UNDEF
CF	4.625971	42	0.01	66.8516	3.809523810	21.26	0.352790
File after applying inversion to a sequence of length three							
UCF	5.861390	26	0.01	126.5014	3.444423114	9.64	-0.029396
CF	7.993551	0	0.01	128.5278	3.100490829	1.31	0.004076
File after applying inversion to a sequence of length four							
UCF	7.093955	11	0.01	126.1196	3.18525628	1.39	0.03054
CF	7.994442	0	0.01	129.5602	3.113264427	0.90	-0.010121
File after applying inversion to a sequence of length five							
UCF	7.501193	6	0.01	125.7701	3.053177193	2.81	0.016942
CF	7.994459	0	0.01	129.4536	3.057069934	2.69	-0.015221
File after applying inversion to a sequence of length six							
UCF	7.643370	4	0.01	125.6156	2.982146285	5.08	0.016278
CF	7.994819	0	0.01	129.6460	3.080473610	1.95	-0.013197
File after applying inversion to a sequence of length seven							
UCF	7.684987	3	0.01	125.7893	2.927625264	6.81	0.017291
CF	7.995342	0	0.01	128.3935	3.123328067	0.58	0.008629
File after applying inversion to a sequence of length eight							
UCF	7.704242	3	0.01	125.7250	2.900364753	7.68	0.017971
CF	7.995339	0	0.01	128.7567	3.085348961	1.79	0.007035
File after applying inversion to a sequence of length nine							
UCF	7.652395	4	0.01	125.6860	2.880783260	8.30	0.018829

CF	7.995118	0	0.01	129.3555	3.080315642	1.95	-0.002605
File after applying inversion to a sequence of length ten							
UCF	7.619051	4	0.01	125.6742	2.871568439	8.60	0.019166
CF	7.995107	0	0.01	128.5142	3.075650376	2.10	0.000643
File after applying inversion to a sequence of length eleven							
UCF	7.601424	4	0.01	125.6672	2.867344980	8.73	0.019284
CF	7.995350	0	0.01	128.9370	3.078602620	2.01	0.001203
File after applying inversion to a sequence of length twelve							
UCF	7.592307	5	0.01	125.6633	2.866961029	8.74	0.019326
CF	7.995413	0	0.01	128.6883	3.074231989	2.14	0.001951
File after applying inversion to a sequence of length thirteen							
UCF	7.587984	5	0.01	125.6618	2.866961029	8.74	0.019336
CF	7.995095	0	0.01	128.7596	3.115195274	0.84	0.001691
File after applying inversion to a sequence of length fourteen							
UCF	7.586127	5	0.01	125.6614	2.866961029	8.74	0.019333
CF	7.994706	0	0.01	128.7746	3.110624795	0.99	-0.003590

Table 2.5 Results of ENT tests on Video File

VIDEO FILE							
File type	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
Totally Random	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
UCF	7.606804	4	0.01	112.1045	3.271680939	4.14	0.256775
CF	7.999189	0	0.01	128.7441	3.106007096	1.13	0.009228
Null Bytes Removed File							
UCF	7.905910	1	0.01	114.2856	3.391006983	7.94	0.034345
CF	7.999110	0	0.01	128.8218	3.105272987	1.16	0.011581
Duplicate Bytes Removed File							
UCF	7.908754	1	0.01	114.5160	3.386551745	7.80	0.024494
CF	7.999112	0	0.01	128.8943	3.100440684	1.31	0.005575
File after applying inversion to a sequence of length two							
UCF	0.000000	100	0.01	170.0000	4.000000000	27.32	UNDEF
CF	2.750624	65	0.01	55.1647	3.979848866	26.68	0.522141
File after applying inversion to a sequence of length three							
UCF	5.867239	26	0.01	122.5914	3.616147915	15.11	-0.039728
CF	7.998999	0	0.01	127.8549	3.126183246	0.49	0.003000
File after applying inversion to a sequence of length four							
UCF	7.175432	10	0.01	120.6774	3.481750960	10.83	0.024000
CF	7.998608	0	0.01	129.0290	3.110402920	0.99	0.001690
File after applying inversion to a sequence of length five							
UCF	7.632515	4	0.01	117.8010	3.442665157	9.58	0.025247
CF	7.998608	0	0.01	129.1819	3.102134974	1.26	-0.001877
File after applying inversion to a sequence of length six							
UCF	7.819440	2	0.01	117.2579	3.405027951	8.39	0.028836
CF	7.998868	0	0.01	129.1311	3.099542992	1.34	0.001101
File after applying inversion to a sequence of length seven							
UCF	7.895737	1	0.01	116.7389	3.386051320	7.78	0.029031
CF	7.998979	0	0.01	129.0007	3.102261020	1.25	0.000328

File after applying inversion to a sequence of length eight							
UCF	7.927758	0	0.01	116.1917	3.378110370	7.53	0.026195
CF	7.998940	0	0.01	129.0005	3.099738557	1.33	0.000147
File after applying inversion to a sequence of length nine							
UCF	7.922218	0	0.01	115.0470	3.388968269	7.87	0.025645
CF	7.998994	0	0.01	128.9252	3.103900566	1.20	0.003272
File after applying inversion to a sequence of length ten							
UCF	7.916712	1	0.01	114.6800	3.387829144	7.84	0.024953
CF	7.999123	0	0.01	128.8346	3.106384763	1.12	0.004075
File after applying inversion to a sequence of length eleven							
UCF	7.913144	1	0.01	114.5566	3.387019247	7.81	0.024526
CF	7.999157	0	0.01	128.8086	3.106872675	1.11	0.006091
File after applying inversion to a sequence of length twelve							
UCF	7.911015	1	0.01	114.5274	3.386643928	7.80	0.024467
CF	7.999143	0	0.01	128.8631	3.101282520	1.28	0.005990
File after applying inversion to a sequence of length thirteen							
UCF	7.909797	1	0.01	114.5190	3.386571498	7.80	0.024470
CF	7.999133	0	0.01	128.8448	3.107417541	1.09	0.005895
File after applying inversion to a sequence of length fourteen							
UCF	7.909106	1	0.01	114.5168	3.386551745	7.80	0.024484
CF	7.999190	0	0.01	128.8451	3.101381398	1.28	0.005742

2.3.2 NIST Tests

Table 2.6 Results of NIST tests on Text - Uncompressed File

UNCOMPRESSED TEXT FILE								
TEST	**SF**	**NRF**	**DRF**	**IF N=2**	**IF N=3**	**IF N=4**	**IF N=5**	**IF N=6**
Approx. Entropy	F	F	F	F	F	F	F	F
Block Frequency	S	S	S	S	S	S	S	S
Cumulative Sum	F	F	F	S	F	S	F	F
FFT	F	F	F	F	S	F	F	F
Frequency	F	F	F	S	F	S	F	F
Linear Complexity	S	S	S	F	S	S	S	S
Longest Run	S	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S	S
Overlapping Template	F	F	F	F	F	F	F	F
Rank	F	F	F	F	S	F	F	F
Runs	F	F	F	F	F	F	F	F
Serial1	F	F	F	F	F	F	F	F
Serial2	F	F	F	F	F	F	F	F
Universal	S	S	S	S	S	S	S	S

Table 2.7 Results of NIST tests on Text - Uncompressed File

UNCOMPRESSED TEXT FILE								
TEST	**IF N=7**	**IF N=8**	**IF N=9**	**IF N=10**	**IF N=11**	**IF N=12**	**IF N=13**	**IF N=14**
Approx. Entropy	F	F	F	F	F	F	F	F
Block Frequency	S	S	S	S	S	S	S	S
Cumulative Sum	F	F	F	F	F	F	F	F

FFT	F	F	F	F	F	F	F	F
Frequency	F	F	F	F	F	F	F	F
Linear Complexity	S	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S	S
Overlapping Template	F	F	F	F	F	F	F	F
Rank	F	F	F	F	F	F	F	F
Runs	F	F	F	F	F	F	F	F
Serial1	F	F	F	F	F	F	F	F
Serial2	F	F	F	F	F	F	F	F
Universal	S	S	S	S	S	S	S	S

Table 2.8 Results of NIST tests on Text – Compressed File

COMPRESSED TEXT FILE								
TEST	**SF**	**NRF**	**DRF**	**IF N=2**	**IF N=3**	**IF N=4**	**IF N=5**	**IF N=6**
Approx. Entropy	F	F	F	-	F	F	F	F
Block Frequency	F	F	F	-	F	F	F	F
Cumulative Sum	S	S	S	-	S	S	S	F
FFT	S	S	S	-	S	S	S	S
Frequency	S	S	S	-	S	S	S	F
Linear Complexity	S	S	S	-	S	S	S	S
Longest Run	S	S	S	-	S	S	S	S
Non-periodic template	S	S	S	-	S	S	S	S
Overlapping Template	S	S	S	-	S	S	S	S
Rank	S	S	S	-	S	S	S	S
Runs	S	S	S	-	S	S	S	S
Serial1	F	F	F	-	F	F	F	F
Serial2	F	F	S	-	F	F	F	F
Universal	S	S	S	-	S	S	S	S

Table 2.9 Results of NIST tests on Text - Compressed File

COMPRESSED TEXT FILE								
TEST	**IF N=7**	**IF N=8**	**IF N=9**	**IF N=10**	**IF N=11**	**IF N=12**	**IF N=13**	**IF N=14**
Approx. Entropy	F	F	F	F	F	F	F	F
Block Frequency	F	F	F	F	F	F	F	F
Cumulative Sum	S	S	S	S	S	S	S	S
FFT	S	S	S	S	S	S	S	S
Frequency	S	S	S	S	S	S	S	S
Linear Complexity	S	F	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S	S	S
Rank	S	S	S	S	S	S	S	S
Runs	S	S	S	S	S	S	S	S
Serial1	F	F	F	F	F	F	F	F
Serial2	S	S	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S	S

2.4 Advantages and Limitations

This algorithm is an inexpensive alternative to shift registers and can be used to generate very long pseudorandom sequences that can be used encrypt even an audio or video file.

The algorithm mainly reduces the problem of large size key distribution to the authorized receiver. One just needs to tell the authorized receiver (in a secure way), the link for source file download (for example) with which the data file is encrypted and the secret value n. The authorized receiver can generate the key file at his own site using the same source file and decrypt the data file from it.

An unauthorized receiver (intruder) cannot generate key file as he has no knowledge of source file. Moreover the brute force attack on source file using the file database is not possible as the file database is infinite. Also a brute force attack for the key file bits is not possible as the size of the key file is very large.

This algorithm makes use of the standard compression software from Microsoft Corporation. It provides optimum compression but still the key file generated from this algorithm does not truly pass the ENT and NIST tests. Hence some other compression mechanism or some encryption mechanism should be used to get the better result.

Moreover this algorithm does not make use of secret key; hence if the source file is compromised in any way then the key file is also not secret any more.

Also, the source files downloaded by the authorized sender and receiver must be identical. To ensure this, the authorized sender and receiver systems can be synchronized.

The seed information, i.e. the source file link and the secret value, must be transmitted to an authorized receiver in a secure way and thus rely on some other encryption techniques like RSA or AES. Hence the pseudorandom key based stream ciphers do not work completely independent of other algorithms and their security may be affected by any attack on the security of these algorithms.

CHAPTER 3

INVERSION-ENCRYPTION METHOD

3.1 Introduction

This is the second proposed algorithm and is an encryption based algorithm. The authorized sender and receiver can share their very large size data over an unsecure channel at a very low cost using this algorithm. For this, the authorized sender and the authorized receiver share a link on the Internet from where a file can be downloaded. This link must be shared using a very secure form of communication so that an unauthorized receiver is not able to see it. The authorized sender and receiver also decide the cryptographic symmetric key algorithm to be used during the process and share the secret key and the secret value n between them through the same secure channel. Then they download the file from the Internet using this link, which is termed as the source file. Then both the authorized sender and receiver generate the key file using this source file. This key file is XORed with the plain data file to generate a cipher data file. Then this cipher data file is transmitted over the unsecure channel.

The information that goes from the authorized sender to the authorized receiver through the secure channel or medium includes: the file download link, the encryption algorithm, the secret key and the secret value n.

The information that goes from the authorized sender to the authorized receiver through the insecure channel or medium includes: the cipher data file encrypted using the key file generated from the proposed method.

3.2 Methodology

The key generation algorithm consists of the following steps.

Step 1: Input the source binary file in binary read mode. Open two other binary files, say X and Y in binary write mode.

Step 2: Segment the source file in one byte segments each.

Step 3: Remove those bytes from the source file whose ASCII value is either 0 or 255. Store the result in X.

Step 4: Close X.

Step 5: Open X in binary read mode.

Step 6: Remove all those bytes from X, which duplicate to their immediate predecessor bytes and store the resultant bytes in Y.

Step 7: Close X and Y.

Step 8: Open X in binary write mode and Y in binary read mode.

Step 9: Scan Y, bit by bit, to get a sequence of n consecutive zeros or n consecutive ones, where $3 \leq n \leq 14$. Note that there will be no sequence of 15 or more consecutive ones or zeros as null bytes are already removed from the file. On getting such a sequence, invert the final bit of the sequence. As a result, there will be no sequence in the file, which is composed of more than n-1 consecutive zeros or n-1 consecutive ones.

Step 10: Store the resultant bytes in X.

Step 11: Close X and Y.

Step 12: Open X in binary read mode and Y in binary write mode.

Step 13: Segment X in 64-bit blocks for DES and IDEA or 128-bit blocks for AES.

Step 14: Encrypt X using some standard secret key encryption algorithm like IDEA, DES or AES in ECB mode. This increases randomness of the file. Also the secret key is only available with the authorized sender and receiver. Hence only the authorized sender and receiver can generate the file.

Step 15: Store the resultant bytes in Y. The resultant file Y serves as the key file during encryption.

Step 16: Close all files.

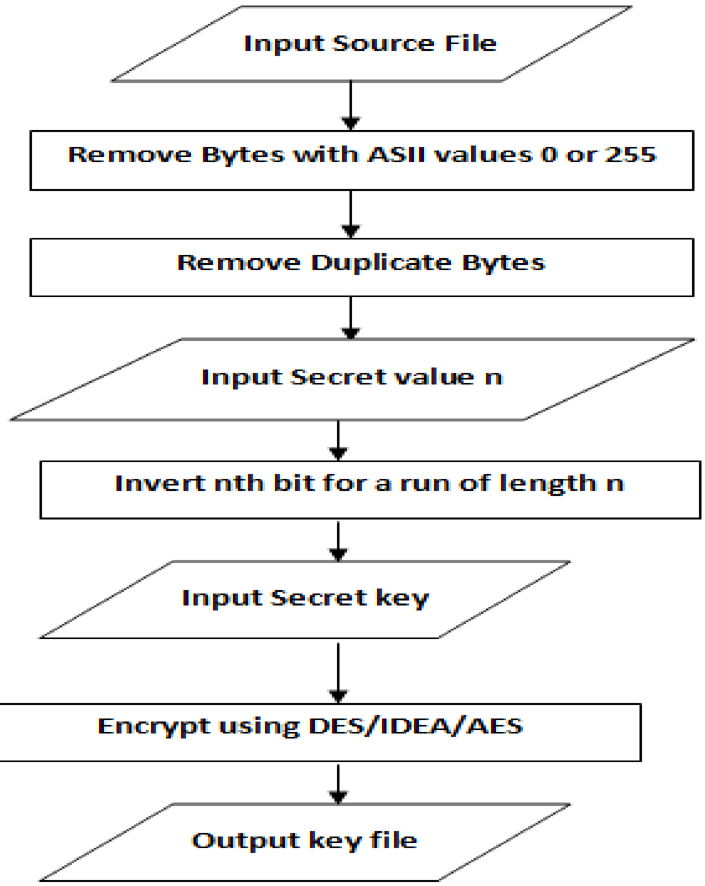

Fig. 3.1 Flowchart for the Inversion-Encryption method

The significance of the major steps of the algorithm is given as follows:

Inversion: In this step, the source file is scanned, bit by bit, to get a sequence of n consecutive zeros or n consecutive ones, where $3 \leq N \leq 14$. On getting such a sequence, the final bit of the sequence is inverted. As a result, there will be no sequence in the file, which is composed of more than N-1 consecutive zeros or N-1 consecutive ones. This N is kept secret. There will be no sequence of 15 or more consecutive ones or zeros as the bytes with ASCII values 0 or 255 are already removed from the file. Inversion for N<3 is not useful at all. Inversion for N=1 corresponds to the inversion of all the bits of the file. Inversion for N=2 corresponds to a file having alternate 1 and 0 bits, which has zero randomness.

Encryption: The results of NIST [32] and ENT [31] tests show that after encryption the randomness of the file increases to a great extent. The randomness of an encrypted file is close to the randomness of a true random file. Also the secret key is only available with the authorized sender and receiver. Hence only the authorized sender and receiver can generate the file. The encryption is performed in ECB [56] mode because in ECB mode, only secret key is used and the initial vector is not required. Hence less information is required to be shared through the secure channel and also the ECB mode is simpler.

3.3 Results

The proposed algorithm was implemented in C and Java programming language to generate the pseudorandom binary key files from various types of source files (i.e. from the text, audio, video and image files). The change in the size of the files after each step is given in table 3.1. We also performed ENT [31] and NIST [32] statistical tests on the source files and the corresponding output files of the key generation algorithm to find the amount of randomness in the files. The ENT and the NIST tests are performed using the standard testing software available online on the official websites of these tests. The results of the ENT tests are given in the tables 3.2, 3.3, 3.4 and 3.5 for text, image, audio and video files respectively. The results of NIST tests for the text file are given in tables 3.6, 3.7, 3.8 and 3.9. The results of NIST tests for the audio file are given in tables 3.10, 3.11, 3.12 and 3.13.

Table 3.1 Size of the files after each step

File Type	Size of SF	Size of NRF	Size of DRF	Size of KF
Text	11.8 KB	11.8 KB	11.6 KB	11.6 KB
Audio	94.7 KB	61.3 KB	61 KB	61 KB
Video	3.97 MB	3.50 MB	3.47 MB	3.47 MB
Image	63.7 KB	61.5 KB	58 KB	58 KB

3.3.1 ENT Tests

Table 3.2 Results of ENT tests on Text File

TEXT FILE							
Encryption type	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	4.364403	45	0.01	93.5753	4.000000000	27.32	0.005331
IDEA	7.982407	0	4.94	126.9986	3.181862987	1.28	-0.010872
Null Bytes Removed File							
NONE	4.364403	45	0.01	93.5753	4.000000000	27.32	0.005331
IDEA	7.982651	0	4.45	126.6336	3.152291769	0.34	0.010005
Duplicate Bytes Removed File							
NONE	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
IDEA	7.983475	0	23.51	126.7933	3.183467742	1.33	-0.028721
File after applying inversion to a sequence of length two							
NONE	0.000000	100	0.01	85.000	4.000000000	27.32	UNDEF
IDEA	3.000000	62	0.01	129.2500	4.000000000	27.32	0.271466
File after applying inversion to a sequence of length three							
NONE	4.021938	49	0.01	136.0150	3.379032258	7.56	-0.207247
IDEA	7.983521	0	26.42	126.7744	3.141129032	0.01	-0.001412
File after applying inversion to a sequence of length four							
NONE	4.313830	46	0.01	101.0115	3.961693548	26.10	-0.064890
IDEA	7.982166	0	4.49	127.1697	3.153225806	0.37	0.019027
File after applying inversion to a sequence of length five							
NONE	4.511224	43	0.01	97.3783	3.870967742	23.22	0.030595
IDEA	7.983702	0	29.97	126.6097	3.153225806	0.37	-0.008499
File after applying inversion to a sequence of length six							
NONE	4.882123	38	0.01	113.4824	3.397177419	8.14	-0.410593
IDEA	7.980907	0	0.80	128.7487	3.151209677	0.31	-0.014041
File after applying inversion to a sequence of length seven							
NONE	4.364658	45	0.01	93.9212	4.000000000	27.32	-0.045451
IDEA	7.983319	0	21.59	126.9124	3.179435484	1.20	-0.028915
File after applying inversion to a sequence of length eight							
NONE	4.372235	45	0.01	93.8406	4.000000000	27.32	-0.035130

IDEA	7.983233	0	18.88	126.7508	3.183467742	1.33	-0.029151
File after applying inversion to a sequence of length nine							
NONE	4.372949	45	0.01	93.8379	4.000000000	27.32	-0.034668
IDEA	7.983423	0	22.73	126.7556	3.183467742	1.33	-0.029203
File after applying inversion to a sequence of length ten							
NONE	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
IDEA	7.983475	0	23.51	126.7933	3.183467742	1.33	-0.028721
File after applying inversion to a sequence of length eleven							
NONE	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
IDEA	7.983475	0	23.51	126.7933	3.183467742	1.33	-0.028721
File after applying inversion to a sequence of length twelve							
NONE	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
IDEA	7.983475	0	23.51	126.7933	3.183467742	1.33	-0.028721
File after applying inversion to a sequence of length thirteen							
NONE	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
IDEA	7.983475	0	23.51	126.7933	3.183467742	1.33	-0.028721
File after applying inversion to a sequence of length fourteen							
NONE	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
IDEA	7.983475	0	23.51	126.7933	3.183467742	1.33	-0.028721

Table 3.3 Results of ENT tests on Image File

IMAGE FILE							
Encryption type	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
Totally Random	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	7.184481	10	0.01	126.8271	3.504828474	11.56	0.435878
IDEA	7.997344	0	73.18	127.0432	3.140544518	0.03	0.002336
Null Bytes Removed File							
NONE	7.190326	10	0.01	124.0478	3.596536302	14.48	0.573973
IDEA	7.996776	0	12.41	127.2197	3.138941759	0.08	0.001124
Duplicate Bytes Removed File							
NONE	7.213525	9	0.01	123.9359	3.612877182	15.00	0.548074
IDEA	7.996417	0	4.32	127.5707	3.116269681	0.81	0.001858
File after applying inversion to a sequence of length two							
NONE	0.000000	100	0.01	85.0000	4.000000000	27.32	UNDEF
IDEA	3.000000	62	0.01	129.2500	4.000000000	27.32	0.271466
File after applying inversion to a sequence of length three							
NONE	5.660456	29	0.01	126.9927	3.795741245	20.82	0.100742
IDEA	7.996730	0	24.69	127.0205	3.167945095	0.84	0.007903
File after applying inversion to a sequence of length four							
NONE	6.770797	15	0.01	125.7696	3.649207791	16.16	0.381586
IDEA	7.997146	0	80.37	127.5064	3.118691966	0.73	0.001749
File after applying inversion to a sequence of length five							
NONE	7.133488	10	0.01	125.4563	3.603189020	14.69	0.459713
IDEA	7.996694	0	22.96	127.8550	3.122325394	0.61	0.001791
File after applying inversion to a sequence of length six							
NONE	7.251777	9	0.01	123.8837	3.612069836	14.98	0.506082
IDEA	7.996770	0	31.06	127.6003	3.119095680	0.72	-0.001814

File after applying inversion to a sequence of length seven							
NONE	7.250141	9	0.01	124.3192	3.624987385	15.39	0.545376
IDEA	7.996820	0	36.69	127.9715	3.111425111	0.96	0.001945
File after applying inversion to a sequence of length eight							
NONE	7.270602	9	0.01	124.0301	3.594711878	14.42	0.538111
IDEA	7.996570	0	11.58	127.3089	3.127169964	0.46	0.003827
File after applying inversion to a sequence of length nine							
NONE	7.233893	9	0.01	123.9347	3.614088203	15.04	0.550350
IDEA	7.996437	0	4.85	127.5997	3.121114251	0.65	-0.000876
File after applying inversion to a sequence of length ten							
NONE	7.225757	9	0.01	123.9300	3.613684529	15.03	0.549011
IDEA	7.996387	0	3.39	127.4927	3.118691966	0.73	-0.001249
File after applying inversion to a sequence of length eleven							
NONE	7.219178	9	0.01	123.9324	3.613684529	15.03	0.548359
IDEA	7.996429	0	4.76	127.5651	3.117884538	0.75	0.001042
File after applying inversion to a sequence of length twelve							
NONE	7.214809	9	0.01	123.9353	3.612877182	15.00	0.548112
IDEA	7.996468	0	6.11	127.5297	3.116269681	0.81	0.001417
File after applying inversion to a sequence of length thirteen							
NONE	7.213700	9	0.01	123.9358	3.612877182	15.00	0.548083
IDEA	7.996416	0	4.30	127.5390	3.117884538	0.75	0.001633
File after applying inversion to a sequence of length fourteen							
NONE	7.213525	9	0.01	123.9359	3.612877182	15.00	0.548074
IDEA	7.996417	0	4.32	127.5707	3.116269681	0.81	0.001858

Table 3.4 Results of ENT tests on Audio File

AUDIO FILE							
Encryption type	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	6.200233	22	0.01	125.2086	2.295751129	26.92	0.174946
IDEA	7.997502	0	0.04	127.4790	3.142998516	0.04	0.001645
Null Bytes Removed File							
NONE	7.580974	5	0.01	125.6436	2.861509074	8.92	0.026084
IDEA	7.996578	0	3.72	127.5801	3.139172796	0.08	-0.005006
Duplicate Bytes Removed File							
NONE	7.585443	5	0.01	125.6610	2.866961029	8.74	0.019331
IDEA	7.996756	0	13.60	127.6824	3.144475377	0.09	-0.004740
File after applying inversion to a sequence of length two							
NONE	0.000000	100	0.01	85.0000	4.000000000	27.32	UNDEF
IDEA	3.000000	62	0.01	129.2500	4.000000000	27.32	0.271466
File after applying inversion to a sequence of length three							
NONE	5.861390	26	0.01	126.5014	3.444423114	9.64	-0.029396
IDEA	7.997054	0	47.15	127.0106	3.140635500	0.03	-0.004090
File after applying inversion to a sequence of length four							
NONE	7.093955	11	0.01	126.1196	3.18525628	1.39	0.03054
IDEA	7.996936	0	30.16	127.3891	3.146011328	0.14	0.003265
File after applying inversion to a sequence of length five							

NONE	7.501193	6	0.01	125.7701	3.053177193	2.81	0.016942
IDEA	7.997280	0	79.70	127.3381	3.131419795	0.32	-0.000602
File after applying inversion to a sequence of length six							
NONE	7.643370	4	0.01	125.6156	2.982146285	5.08	0.016278
IDEA	7.996957	0	35.99	127.2388	3.145627340	0.13	0.005285
File after applying inversion to a sequence of length seven							
NONE	7.684987	3	0.01	125.7893	2.927625264	6.81	0.017291
IDEA	7.996723	0	10.49	127.5751	3.120668139	0.67	-0.001743
File after applying inversion to a sequence of length eight							
NONE	7.704242	3	0.01	125.7250	2.900364753	7.68	0.017971
IDEA	7.996449	0	1.37	127.7462	3.124508016	0.54	-0.003530
File after applying inversion to a sequence of length nine							
NONE	7.652395	4	0.01	125.6860	2.880783260	8.30	0.018829
IDEA	7.997021	0	42.59	127.4060	3.141019487	0.02	-0.004650
File after applying inversion to a sequence of length ten							
NONE	7.619051	4	0.01	125.6742	2.871568439	8.60	0.019166
IDEA	7.996839	0	20.02	127.4175	3.151387156	0.31	-0.004831
File after applying inversion to a sequence of length eleven							
NONE	7.601424	4	0.01	125.6672	2.867344980	8.73	0.019284
IDEA	7.996848	0	21.15	127.6737	3.137563598	0.13	-0.004927
File after applying inversion to a sequence of length twelve							
NONE	7.592307	5	0.01	125.6633	2.866961029	8.74	0.019326
IDEA	7.996782	0	15.93	127.5377	3.145243352	0.12	-0.004949
File after applying inversion to a sequence of length thirteen							
NONE	7.587984	5	0.01	125.6618	2.866961029	8.74	0.019336
IDEA	7.996741	0	12.71	127.6942	3.143323414	0.06	-0.005023
File after applying inversion to a sequence of length fourteen							
NONE	7.586127	5	0.01	125.6614	2.866961029	8.74	0.019333
IDEA	7.996761	0	14.04	127.6765	3.142171451	0.02	-0.004876

Table 3.5 Results of ENT tests on Video File

VIDEO FILE							
Encryption type	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	7.606804	4	0.01	112.1045	3.271680939	4.14	0.256775
IDEA	7.935663	0	0.01	125.0358	3.176128177	1.10	-0.010936
Null Bytes Removed File							
NONE	7.905910	1	0.01	114.2856	3.391006983	7.94	0.034345
IDEA	7.999931	0	0.01	127.5083	3.142984869	0.04	-0.002135
Duplicate Bytes Removed File							
NONE	7.908754	1	0.01	114.5160	3.386551745	7.80	0.024494
IDEA	7.999934	0	0.08	127.4725	3.141887332	0.01	0.000433
File after applying inversion to a sequence of length two							
NONE	0.000000	100	0.01	170.0000	4.000000000	27.32	UNDEF
IDEA	3.000000	62	0.01	186.3750	1.000003292	68.17	-0.414491
File after applying inversion to a sequence of length three							
NONE	5.867239	26	0.01	122.5914	3.616147915	15.11	-0.039728

IDEA	7.999939	0	1.38	127.4885	3.142624802	0.03	-0.000481
File after applying inversion to a sequence of length four							
NONE	7.175432	10	0.01	120.6774	3.481750960	10.83	0.024000
IDEA	7.999945	0	14.91	127.5084	3.142657725	0.03	0.001103
File after applying inversion to a sequence of length five							
NONE	7.632515	4	0.01	117.8010	3.442665157	9.58	0.025247
IDEA	7.999939	0	1.44	127.5310	3.142572126	0.03	0.000066
File after applying inversion to a sequence of length six							
NONE	7.819440	2	0.01	117.2579	3.405027951	8.39	0.028836
IDEA	7.999941	0	3.74	127.4779	3.140425560	0.04	-0.000785
File after applying inversion to a sequence of length seven							
NONE	7.895737	1	0.01	116.7389	3.386051320	7.78	0.029031
IDEA	7.999940	0	1.96	127.5087	3.142624802	0.03	-0.001025
File after applying inversion to a sequence of length eight							
NONE	7.927758	0	0.01	116.1917	3.378110370	7.53	0.026195
IDEA	7.999937	0	0.36	127.5299	3.141288137	0.01	0.000371
File after applying inversion to a sequence of length nine							
NONE	7.922218	0	0.01	115.0470	3.388968269	7.87	0.025645
IDEA	7.999942	0	4.98	127.5018	3.141558104	0.00	0.000032
File after applying inversion to a sequence of length ten							
NONE	7.916712	1	0.01	114.6800	3.387829144	7.84	0.024953
IDEA	7.999942	0	4.25	127.4749	3.142177052	0.02	0.000416
File after applying inversion to a sequence of length eleven							
NONE	7.913144	1	0.01	114.5566	3.387019247	7.81	0.024526
IDEA	7.999936	0	0.27	127.4640	3.141801732	0.01	0.000324
File after applying inversion to a sequence of length twelve							
NONE	7.911015	1	0.01	114.5274	3.386643928	7.80	0.024467
IDEA	7.999935	0	0.17	127.4758	3.141274968	0.01	0.000109
File after applying inversion to a sequence of length thirteen							
NONE	7.909797	1	0.01	114.5190	3.386571498	7.80	0.024470
IDEA	7.999934	0	0.06	127.4795	3.141834655	0.01	0.000466
File after applying inversion to a sequence of length fourteen							
NONE	7.909106	1	0.01	114.5168	3.386551745	7.80	0.024484
IDEA	7.999934	0	0.09	127.4736	3.142078284	0.02	0.000433

3.3.2 NIST Tests

Table 3.6 Results of NIST tests on Text - Unencrypted File

UNENCRYPTED TEXT FILE								
TEST	**SF**	**NRF**	**DRF**	**IF N=2**	**IF N=3**	**IF N=4**	**IF N=5**	**IF N=6**
Approx. Entropy	F	F	F	F	F	F	F	F
Block Frequency	S	S	S	S	S	S	S	S
Cumulative Sum	F	F	F	S	F	S	F	F
FFT	F	F	F	F	S	F	F	F
Frequency	F	F	F	S	F	S	F	F
Linear Complexity	S	S	S	F	S	S	S	S
Longest Run	S	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S	S
Overlapping Template	F	F	F	F	F	F	F	F

Rank	F	F	F	F	S	F	F	F
Runs	F	F	F	F	F	F	F	F
Serial1	F	F	F	F	F	F	F	F
Serial2	F	F	F	F	F	F	F	F
Universal	S	S	S	S	S	S	S	S

Table 3.7 Results of NIST tests on Text - Unencrypted File

UNENCRYPTED TEXT FILE								
TEST	**IF N=7**	**IF N=8**	**IF N=9**	**IF N=10**	**IF N=11**	**IF N=12**	**IF N=13**	**IF N=14**
Approx. Entropy	F	F	F	F	F	F	F	F
Block Frequency	S	S	S	S	S	S	S	S
Cumulative Sum	F	F	F	F	F	F	F	F
FFT	F	F	F	F	F	F	F	F
Frequency	F	F	F	F	F	F	F	F
Linear Complexity	S	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S	S
Overlapping Template	F	F	F	F	F	F	F	F
Rank	F	F	F	F	F	F	F	F
Runs	F	F	F	F	F	F	F	F
Serial1	F	F	F	F	F	F	F	F
Serial2	F	F	F	F	F	F	F	F
Universal	S	S	S	S	S	S	S	S

Table 3.8 Results of NIST tests on Text - Encrypted File

IDEA ENCRYPTED TEXT FILE								
TEST	**SF**	**NRF**	**DRF**	**IF N=2**	**IF N=3**	**IF N=4**	**IF N=5**	**IF N=6**
Approx. Entropy	S	S	S	F	S	S	S	S
Block Frequency	S	S	S	S	S	S	S	S
Cumulative Sum	S	S	S	F	S	S	S	S
FFT	S	S	S	F	S	S	S	S
Frequency	S	S	S	F	S	S	S	S
Linear Complexity	S	F	S	F	S	S	S	S
Longest Run	S	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S	S
Overlapping Template	S	S	S	F	S	S	S	S
Rank	S	S	S	F	S	S	S	S
Runs	S	S	S	F	S	S	S	S
Serial1	S	S	S	F	S	S	S	S
Serial2	S	S	S	F	S	S	S	S
Universal	S	S	S	S	S	S	S	S

Table 3.9 Results of NIST tests on Text - Encrypted File

IDEA ENCRYPTED TEXT FILE								
TEST	**IF N=7**	**IF N=8**	**IF N=9**	**IF N=10**	**IF N=11**	**IF N=12**	**IF N=13**	**IF N=14**
Approx. Entropy	S	S	S	S	S	S	S	S
Block Frequency	S	S	S	S	S	S	S	S
Cumulative Sum	S	S	S	S	S	S	S	S
FFT	S	S	S	S	S	S	S	S
Frequency	S	S	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S	S	S
Rank	S	S	S	S	S	S	S	S
Runs	S	S	S	S	S	S	S	S
Serial1	S	S	S	S	S	S	S	S
Serial2	S	S	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S	S

Table 3.10 Results of NIST tests on Audio - Unencrypted File

UNENCRYPTED AUDIO FILE								
TEST	**SF**	**NRF**	**DRF**	**IF N=2**	**IF N=3**	**IF N=4**	**IF N=5**	**IF N=6**
Approx. Entropy	F	F	F	F	F	F	F	F
Block Frequency	F	F	F	S	S	F	F	F
Cumulative Sum	F	F	F	S	S	S	F	F
FFT	F	S	S	F	F	F	S	S
Frequency	F	F	F	S	S	S	S	S
Linear Complexity	S	S	F	F	S	S	S	S
Longest Run	S	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S	S
Overlapping Template	F	S	S	F	F	F	F	F
Rank	F	S	S	F	S	S	S	S
Runs	F	F	F	F	F	F	S	F
Serial1	F	F	F	F	F	F	F	F
Serial2	F	F	S	F	F	F	F	F
Universal	S	S	S	S	S	S	S	S

Table 3.11 Results of NIST tests on Audio - Unencrypted File

UNENCRYPTED AUDIO FILE								
TEST	**IF N=7**	**IF N=8**	**IF N=9**	**IF N=10**	**IF N=11**	**IF N=12**	**IF N=13**	**IF N=14**
Approx. Entropy	F	F	F	F	F	F	F	F
Block Frequency	F	F	F	F	F	F	F	F
Cumulative Sum	F	F	F	F	F	F	F	F
FFT	S	S	S	S	S	S	S	S

Frequency	F	S	F	F	F	F	F	F
Linear Complexity	S	S	S	F	S	F	F	F
Longest Run	S	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S	S
Overlapping Template	F	F	F	S	S	S	S	S
Rank	S	S	S	S	S	S	S	S
Runs	F	F	F	F	F	F	F	F
Serial1	F	F	F	F	F	F	F	F
Serial2	F	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S	S

Table 3.12 Results of NIST tests on Audio - Encrypted File

IDEA ENCRYPTED AUDIO FILE								
TEST	**SF**	**NRF**	**DRF**	**IF N=2**	**IF N=3**	**IF N=4**	**IF N=5**	**IF N=6**
Approx. Entropy	S	S	S	F	S	S	S	S
Block Frequency	S	S	S	S	S	S	S	S
Cumulative Sum	S	S	S	F	S	S	S	S
FFT	S	S	S	F	S	S	S	S
Frequency	S	S	S	F	S	S	S	S
Linear Complexity	S	S	S	F	S	S	S	S
Longest Run	S	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S	S
Overlapping Template	S	S	S	F	S	S	S	S
Rank	S	S	S	F	S	S	S	S
Runs	S	S	S	F	S	S	S	S
Serial1	S	S	S	F	S	S	S	S
Serial2	S	S	S	F	S	S	S	S
Universal	S	S	S	S	S	S	S	S

Table 3.13 Results of NIST tests on Audio - Encrypted File

IDEA ENCRYPTED AUDIO FILE								
TEST	**IF N=7**	**IF N=8**	**IF N=9**	**IF N=10**	**IF N=11**	**IF N=12**	**IF N=13**	**IF N=14**
Approx. Entropy	S	S	S	S	S	S	S	S
Block Frequency	S	S	S	S	S	S	S	S
Cumulative Sum	S	S	S	S	S	S	S	S
FFT	S	S	S	S	S	S	S	S
Frequency	S	S	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S	S	S
Rank	S	S	S	S	S	S	S	S
Runs	S	S	S	S	S	S	S	S
Serial1	S	S	S	S	S	S	S	S
Serial2	S	S	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S	S

3.4 Advantages and Limitations

The algorithm mainly reduces the problem of large size key distribution to the authorized receiver. One just needs to tell the authorized receiver (in a secure way), the song or movie name (for example) with which the data file is encrypted, the value of n, the encryption algorithm and the secret key. The authorized receiver can generate the key file at his own site using the same popular song or movie file and decrypt the data file from it. An unauthorized receiver (intruder) does not have the knowledge of which song or movie file is used as key or even the extension of the file used for key generation and hence cannot generate key file. Even if the unauthorized receiver knows the source file from which the key file is generated, he cannot generate the key file as he does not know the secret key used at the encryption stage. Moreover the brute force attack on source file using the file database is not possible as the file database is infinite. Also a brute force attack on the source file bits is not possible as the size of the source file is very large.

This algorithm can be used to generate key file from a source file having any kind of file extension and can be used to encrypt a file of type of extension.

The algorithm is simple and executes faster as it can be implemented in C programming language. The algorithm attempts to reduce the redundancy in the source file to a much greater extent.

This algorithm does not use initial vector and hence only one security weapon is with the authorized sender and receiver i.e. the secret key.

Also, the source files downloaded by the authorized sender and receiver must be identical. To ensure this, the authorized sender and receiver systems can be synchronized.

The seed information, i.e. the source file link, the secret key and the secret value, must be transmitted to an authorized receiver in a secure way and thus rely on some other encryption techniques like RSA or AES. Hence the pseudorandom key based stream ciphers do not work completely independent of other algorithms and their security may be affected by any attack on the security of these algorithms.

CHAPTER 4

DUPLICATE BLOCKS REMOVAL-
ECB MODE ENCRYPTION METHOD

4.1 Introduction

This is the third proposed algorithm and is an encryption based algorithm. The authorized sender and receiver can share their very large size data over an unsecure channel at a very low cost using this algorithm. For this, the authorized sender and the authorized receiver share a link on the Internet from where a file can be downloaded. This link must be shared using a very secure form of communication so that an unauthorized receiver is not able to see it. The authorized sender and receiver also decide the cryptographic symmetric key algorithm to be used during the process and share the secret key between them through the same secure channel. Then they download the file from the Internet using this link, which is termed as the source file. Then both the authorized sender and receiver generate the key file using this source file. This key file is XORed with the plain data file to generate a cipher data file. Then this cipher data file is transmitted over the unsecure channel.

The information that goes from the authorized sender to the authorized receiver through the secure channel or medium includes: the file download link, the encryption algorithm and the secret key.

The information that goes from the authorized sender to the authorized receiver through the insecure channel or medium includes: the cipher data file encrypted using the key file generated from the proposed method.

4.2 Methodology

The key generation algorithm consists of the following steps.

Step 1: Open the source file in binary read mode. Open two other binary files, say X and Y in binary write mode.

Step 2: Remove the header from the source file. Store the result in X. Also, instead of generating the key from the whole file, we can also generate the key from a segment of the source file. This will be useful when the size of key is required to be very small as compared to the size of source file. This will speed up the process.

Step 3: Close all X.

Step 4: Open X in binary read mode.

Step 5: Segment X in 64-bit blocks for DES and IDEA or 128-bit blocks for AES.

Step 6: Encrypt X using some standard secret key encryption algorithm like IDEA, DES or AES in ECB mode. Remove all successive duplicate blocks from the file or alternatively remove all duplicate bytes which occur eight times in a sequence from the file. This increases randomness of the file. Also the secret key is only available with the authorized sender and receiver. Hence only the authorized sender and receiver can generate the file.

Step 7: Store the resultant bytes in Y. The resultant file Y serves as the key file during encryption.

Step 8: Close all files.

The significance of the major steps of the algorithm is given as follows:

Header Removal: The header is removed as the header of all files with same extension will be almost similar. Thus, if not removed, a cryptanalyst can easily predict the first few bytes of the key sequence. Also, instead of generating the key from the whole file, we can also generate the key from a segment of the source file. This will be useful when the size of key is required to be very small as compared to the size of source file. This will speed up the process.

Encryption: The results of NIST [32] and ENT [31] tests show that after encryption the randomness of the file increases to a great extent. The randomness of an encrypted file is close to the randomness of a true random file. Also the secret key is only available with the authorized sender and receiver. Hence only the authorized sender and receiver can generate the file. The encryption is performed in ECB [56] mode because in ECB mode, only secret key is used and the initial vector is not required. Hence less information is required to be shared through the secure channel and also the ECB mode is simpler.

4.3 Results

The proposed algorithm was implemented in C and Java programming language to generate the pseudorandom binary key files from various types of source files (i.e. from the text, audio, video and image files). The change in the size of the files after each step is given in table 4.1. We also performed ENT [31] and NIST [32] statistical tests on the source files and the corresponding output files of the key generation algorithm to find the amount of randomness in the files. The ENT and the NIST tests are performed using the standard testing software available online on the official websites of these tests. The results of the ENT tests are given in the tables 4.2, 4.3, 4.4 and 4.5 for text, image, audio and video files respectively. The results of NIST tests for the text file are given in tables 4.6 and 4.7. The results of NIST tests for the image file are given in tables 4.8 and 4.9. The results of NIST tests for the audio file are given in tables 4.10 and 4.11. The results of NIST tests for the video file are given in tables 4.12 and 4.13.

Table 4.1 Size of the files after each step

File Type	Size of SF	Size of DRF	Size of DBRF	Size of KF
Text	11.8 KB	11.6 KB	11.1 KB	11.1 KB
Audio	94.7 KB	93.6 KB	93.6 KB	93.6 KB
Video	3.97 MB	3.56 MB	3.54 MB	3.54 MB
Image	63.7 KB	60.1 KB	60.1 KB	60.1 KB

4.3.1 ENT Tests

Table 4.2 Results of ENT tests on Text File

TEXT FILE							
Encryption type	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	4.364403	45	0.01	93.5753	4.000000000	27.32	0.005331
DES	7.981000	0	0.21	127.4446	3.116141732	0.81	0.003129
AES	7.981411	0	0.66	126.8760	3.125984252	0.50	-0.003826
IDEA	7.982407	0	4.94	126.9986	3.181862987	1.28	-0.010872
Duplicate Bytes Removed File							
NONE	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
DES	7.984920	0	63.14	127.5600	3.125440806	0.51	-0.001218
AES	7.986449	0	92.84	127.9104	3.198388721	1.81	-0.001904
IDEA	7.983475	0	23.51	126.7933	3.183467742	1.33	-0.028721
Duplicate Blocks Removed File							
NONE	4.374020	45	0.01	93.6546	4.000000000	27.32	-0.038895
DES	7.985385	0	87.59	127.7695	3.107368421	1.09	-0.003982
AES	7.984903	0	75.08	127.5565	3.093108890	1.54	0.006160
IDEA	7.983946	0	55.42	126.9711	3.226146547	2.69	-0.026677

Table 4.3 Results of ENT tests on Image File

IMAGE FILE							
Encryption type	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	7.184481	10	0.01	126.8271	3.504828474	11.56	0.435878
DES	7.996706	0	3.91	127.5251	3.163877138	0.71	0.006217
AES	7.996834	0	8.19	126.9396	3.165348538	0.76	-0.002723
IDEA	7.997344	0	73.18	127.0432	3.140544518	0.03	0.002336
Duplicate Bytes Removed File							
NONE	7.203551	9	0.01	127.0198	3.497174040	11.32	0.403755
DES	7.997040	0	51.53	127.8566	3.129994153	0.37	-0.002361
AES	7.996831	0	23.06	127.2630	3.157053780	0.49	0.001480
IDEA	7.996900	0	32.83	127.9457	3.141604132	0.00	-0.002083
Duplicate Blocks Removed File							
NONE	7.203451	9	0.01	127.0309	3.497125037	11.32	0.403595
DES	7.997046	0	52.44	127.8644	3.129994153	0.37	-0.002380
AES	7.996817	0	21.41	127.2652	3.157833203	0.52	0.001500
IDEA	7.996902	0	33.22	127.9447	3.141520468	0.00	-0.002100

Table 4.4 Results of ENT tests on Audio File

AUDIO FILE							
Encryption type	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	6.200233	22	0.01	125.2086	2.295751129	26.92	0.174946
DES	7.997638	0	0.30	127.6035	3.126035869	0.50	-0.005894
AES	7.998136	0	55.02	127.1001	3.152504638	0.35	-0.000312
IDEA	7.997502	0	0.04	127.4790	3.142998516	0.04	0.001645
Duplicate Bytes Removed File							
NONE	6.238276	22	0.01	125.9610	2.267692885	27.82	0.160875
DES	7.997572	0	0.25	127.8322	3.130772119	0.34	-0.003679
AES	7.998031	0	37.94	127.8142	3.132632633	0.29	0.001870
IDEA	7.998041	0	38.71	127.4127	3.126712972	0.47	0.001975
Duplicate Blocks Removed File							
NONE	6.238962	22	0.01	125.9726	2.267651477	27.82	0.160845
DES	7.997562	0	0.22	127.8418	3.130500094	0.35	-0.003575
AES	7.997994	0	30.91	127.8323	3.131555889	0.32	0.001347
IDEA	7.998058	0	42.78	127.4100	3.128583949	0.41	0.002027

Table 4.5 Results of ENT tests on Video File

VIDEO FILE							
Encryption type	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	7.606804	4	0.01	112.1045	3.271680939	4.14	0.256775
DES	7.941794	0	0.01	127.4130	3.101435687	1.28	-0.015874
AES	7.967535	0	0.01	126.9710	3.167071221	0.81	-0.006830
IDEA	7.935663	0	0.01	125.0358	3.176128177	1.10	-0.010936
Duplicate Bytes Removed File							
NONE	7.892749	1	0.01	112.4897	3.389071855	7.88	0.031485
DES	7.999929	0	0.01	127.5550	3.141130972	0.01	0.000594
AES	7.999955	0	85.58	127.5571	3.137745981	0.12	-0.000750
IDEA	7.999928	0	0.01	127.4735	3.142140192	0.02	-0.000731
Duplicate Blocks Removed File							
NONE	7.897290	1	0.01	112.7841	3.388054407	7.85	0.030452
DES	7.999948	0	26.10	127.5312	3.137057753	0.14	0.000482
AES	7.999950	0	43.56	127.4952	3.141353051	0.01	0.000415
IDEA	7.999948	0	29.81	127.5053	3.145908644	0.14	-0.000751

4.3.2 NIST Tests

Table 4.6 Results of NIST tests on Text – unencrypted and DES encrypted Files

TEXT FILE						
TEST	UNENCRYPTED			DES ENCRYPTED		
	SF	DRF	DBRF	SF	DRF	DBRF
Approx. Entropy	F	F	F	S	S	S
Block Frequency	S	S	S	S	S	S
Cumulative Sum	F	F	F	S	S	S
FFT	F	F	F	S	S	S
Frequency	F	F	F	S	S	S
Linear Complexity	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S
Overlapping Template	F	F	F	S	S	S
Rank	F	F	F	S	S	S
Runs	F	F	F	S	S	S
Serial1	F	F	F	F	S	S
Serial2	F	F	F	S	S	S
Universal	S	S	S	S	S	S

Table 4.7 Results of NIST tests on Text – AES and IDEA encrypted Files

TEXT FILE						
TEST	AES ENCRYPTED			IDEA ENCRYPTED		
	SF	DRF	DBRF	SF	DRF	DBRF
Approx. Entropy	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S
Cumulative Sum	S	S	S	S	S	S
FFT	S	F	S	S	S	S
Frequency	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S
Rank	S	F	S	S	S	S
Runs	S	S	S	S	S	S
Serial1	S	S	S	S	S	S
Serial2	S	S	S	S	S	S
Universal	S	S	S	S	S	S

Table 4.8 Results of NIST tests on Image – unencrypted and DES encrypted Files

IMAGE FILE						
TEST	UNENCRYPTED			DES ENCRYPTED		
	SF	DRF	DBRF	SF	DRF	DBRF
Approx. Entropy	F	F	F	S	S	S
Block Frequency	F	F	F	S	S	S
Cumulative Sum	F	F	F	S	S	S
FFT	F	F	F	S	S	S

Frequency	F	F	F	S	S	S
Linear Complexity	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S
Overlapping Template	F	F	F	S	S	S
Rank	F	F	F	S	S	S
Runs	F	F	F	S	S	S
Serial1	F	F	F	S	S	S
Serial2	F	F	F	S	S	S
Universal	S	S	S	S	S	S

Table 4.9 Results of NIST tests on Image – AES and IDEA encrypted Files

IMAGE FILE						
TEST	**AES ENCRYPTED**			**IDEA ENCRYPTED**		
	SF	**DRF**	**DBRF**	**SF**	**DRF**	**DBRF**
Approx. Entropy	S	S	S	S	S	S
Block Frequency	S	S	S	S	S	S
Cumulative Sum	S	S	S	S	S	S
FFT	S	S	S	S	S	S
Frequency	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S
Rank	S	S	S	S	S	S
Runs	S	S	S	S	S	S
Serial1	S	S	S	S	S	S
Serial2	S	S	S	S	S	S
Universal	S	S	S	S	S	S

Table 4.10 Results of NIST tests on Audio – unencrypted and DES encrypted Files

AUDIO FILE						
TEST	**UNENCRYPTED**			**DES ENCRYPTED**		
	SF	**DRF**	**DBRF**	**SF**	**DRF**	**DBRF**
Approx. Entropy	F	F	F	S	S	S
Block Frequency	F	F	F	S	S	S
Cumulative Sum	F	F	F	S	S	S
FFT	F	F	F	S	S	S
Frequency	F	F	F	S	S	S
Linear Complexity	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S
Overlapping Template	F	F	F	S	S	S
Rank	F	F	F	S	S	S
Runs	F	F	F	S	S	S
Serial1	F	F	F	S	S	S
Serial2	F	F	F	S	S	S
Universal	S	S	S	S	S	S

Table 4.11 Results of NIST tests on Audio – AES and IDEA encrypted Files

AUDIO FILE						
TEST	AES ENCRYPTED			IDEA ENCRYPTED		
	SF	DRF	DBRF	SF	DRF	DBRF
Approx. Entropy	S	S	S	S	S	S
Block Frequency	S	S	S	S	S	S
Cumulative Sum	S	S	S	S	S	S
FFT	S	S	S	S	S	S
Frequency	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S
Rank	S	S	S	S	S	S
Runs	S	S	S	S	S	S
Serial1	S	S	S	S	S	S
Serial2	S	S	S	S	S	S
Universal	S	S	S	S	S	S

Table 4.12 Results of NIST tests on Video – unencrypted and DES encrypted Files

VIDEO FILE						
TEST	UNENCRYPTED			AES ENCRYPTED		
	SF	DRF	DBRF	SF	DRF	DBRF
Approx. Entropy	F	F	F	F	S	S
Block Frequency	F	F	F	S	S	S
Cumulative Sum	F	F	F	F	S	S
FFT	F	F	F	F	S	S
Frequency	F	F	F	F	S	S
Linear Complexity	F	S	S	F	S	S
Longest Run	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S
Rank	F	F	F	F	S	S
Runs	F	F	F	F	S	S
Serial1	F	F	F	F	S	S
Serial2	F	F	F	F	S	S
Universal	S	S	S	S	S	S

Table 4.13 Results of NIST tests on Video – AES and IDEA Files

VIDEO FILE						
TEST	DES ENCRYPTED			IDEA ENCRYPTED		
	SF	DRF	DBRF	SF	DRF	DBRF
Approx. Entropy	F	F	S	F	F	S
Block Frequency	S	S	S	S	S	S
Cumulative Sum	F	S	S	F	S	S
FFT	F	S	S	F	F	S
Frequency	F	S	S	F	S	S
Linear Complexity	F	S	S	F	S	S

Longest Run	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S
Overlapping Template	S	F	S	S	S	S
Rank	F	S	S	F	F	S
Runs	F	F	S	F	S	S
Serial1	F	F	S	F	F	S
Serial2	F	F	S	F	F	S
Universal	S	S	S	S	S	S

4.4 Advantages and Limitations

The algorithm mainly reduces the problem of large size key distribution to the authorized receiver. One just requires to tell the authorized receiver (in a secure way), the song or movie name or link (for example) with which the data file is encrypted, the encryption algorithm and the secret key. The authorized receiver can generate the key file at his own site using the same popular song or movie file and decrypt the data file from it. An unauthorized receiver (intruder) does not have the knowledge of which song or movie file is used as key or even the extension of the file used for key generation and hence cannot generate key file. Even if the unauthorized receiver knows the source file from which the key file is generated, he cannot generate the key file as he does not know the secret key used at the encryption stage. Moreover the brute force attack on source file using the file database is not possible as the file database on the Internet is infinite. Also a brute force attack on the source file bits is not possible as the size of the source file is very large.

This algorithm can be used to generate key file from a source file having any kind of file extension and can be used to encrypt a file of type of extension.

The algorithm attempts to reduce the redundancy in the source file to a much greater extent.

This algorithm also does not use initial vector and hence only one weapon for security is with the authorized sender and receiver i.e. the secret key.

This algorithm may execute slowly. Also, the source files downloaded by the authorized sender and receiver must be identical. To ensure this, the authorized sender and receiver systems can be synchronized.

The seed information, i.e. the source file link and the secret key, must be transmitted to an authorized receiver in a secure way and thus rely on some other encryption techniques like RSA or AES. Hence the pseudorandom key based stream ciphers do not work completely independent of other algorithms and their security may be affected by any attack on the security of these algorithms.

CHAPTER 5

DUPLICATE BLOCKS REMOVAL-CHAINING MODE ENCRYPTION METHOD

5.1 Introduction

This is the fourth proposed algorithm and is an encryption based algorithm. The authorized sender and receiver can share their very large size data over an unsecure channel at a very low cost using this algorithm. For this, the authorized sender and the authorized receiver share a link on the Internet from where a file can be downloaded. This link must be shared using a very secure form of communication so that an unauthorized receiver is not able to see it. The authorized sender and receiver also decide the cryptographic symmetric key algorithm to be used during the process and share the secret key and the initial vector between them through the same secure channel. Then they download the file from the Internet using this link, which is termed as the source file. Then both the authorized sender and receiver generate the key file using this source file. This key file is XORed with the plain data file to generate a cipher data file. Then this cipher data file is transmitted over the unsecure channel.

The information that goes from the authorized sender to the authorized receiver through the secure channel or medium includes: the file download link, the encryption algorithm, the chaining mode, the secret key and the initial vector.

The information that goes from the authorized sender to the authorized receiver through the insecure channel or medium includes: the cipher data file encrypted using the key file generated from the proposed method.

5.2 Methodology

The key generation algorithm consists of the following steps.

Step 1: Open the source file in binary read mode. Open two other binary files, say X and Y in binary write mode.

Step 2: Remove the header from the source file. Store the result in X. Also, instead of generating the key from the whole file, we can also generate the key from a segment of the source file. This will be useful when the size of key is required to be very small as compared to the size of source file. This will speed up the process.

Step 3: Close all X.

Step 4: Open X in binary read mode.

Step 5: Segment X in 64-bit blocks for DES and IDEA or 128-bit blocks for AES.

Step 6: Encrypt X using some standard secret key encryption algorithm like IDEA, DES or AES in chaining mode (CBC/CFB/OFB). Remove all successive duplicate blocks from the file or alternatively remove all duplicate bytes which occur eight times in a sequence from the file. This increases randomness of the file. Also the secret key and initial vector are only available with the authorized sender and receiver. Hence only the authorized sender and receiver can generate the file.

Step 7: Store the resultant bytes in Y. The resultant file Y serves as the key file during encryption.

Step 8: Close all files.

The significance of the major steps of the algorithm is given as follows:

Header Removal: The header is removed as the header of all files with same extension will be almost similar. Thus, if not removed, a cryptanalyst can easily predict the first few bytes of the key sequence. Also, instead of generating the key from the whole file, we can also generate the key from a segment of the source file. This will be useful when the size of key is required to be very small as compared to the size of source file. This will speed up the process.

Encryption: The results of NIST and ENT tests show that after encryption the randomness of the file increases to a great extent. The randomness of an encrypted file is close to the randomness of a true random file. The encryption is performed in chaining mode (CBC/CFB/OFB) [56] and not in

ECB [56] mode because in ECB mode, a codebook attack is possible. In this attack, if two or more blocks of plaintext are identical then the corresponding ciphertext blocks will also be identical. Thus a cryptanalyst can use this information to attack the secret system. It does not matter that the identical blocks of the plaintext reside at which position in the source file. The probability of a codebook attack increases when the number of such identical blocks is large in the source file. Performing the encryption in chaining mode removes the probability of the codebook attack.

5.3 Results

The proposed algorithm was implemented in C and Java programming language to generate the pseudorandom binary key files from various types of source files (i.e. from the text, audio, video and image files). The change in the size of the files after each step is given in table 5.1. We also performed ENT [31] and NIST [32] statistical tests on the source files and the corresponding output files of the key generation algorithm to find the amount of randomness in the files. The ENT and the NIST tests are performed using the standard testing software available online on the official websites of these tests. The results of the ENT tests are given in the tables 5.2, 5.3, 5.4 and 5.5 for text, image, audio and video files respectively. The results of NIST tests for the text file are given in tables 5.6, 5.7 and 5.8. The results of NIST tests for the image file are given in tables 5.9, 5.10 and 5.11. The results of NIST tests for the audio file are given in tables 5.12, 5.13 and 5.14. The results of NIST tests for the video file are given in tables 5.15, 5.16 and 5.17.

Table 5.1 Size of the files after each step

File Type	Size of SF	Size of DRF	Size of DBRF	Size of KF
Text	11.8 KB	11.6 KB	11.1 KB	11.1 KB
Audio	94.7 KB	93.6 KB	93.6 KB	93.6 KB
Video	3.97 MB	3.56 MB	3.54 MB	3.54 MB
Image	63.7 KB	60.1 KB	60.1 KB	60.1 KB

5.3.1 ENT Tests

Table 5.2 Results of ENT tests on Text File

TEXT FILE							
Encryption type/ mode	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	4.364403	45	0.01	93.5753	4.000000000	27.32	0.005331
AES Encrypted Source File							
CBC	7.981414	0	0.85	129.2305	3.059055118	2.63	0.003301
CFB	7.984934	0	48.98	128.0091	3.088582677	1.69	-0.012361
OFB	7.984554	0	35.51	127.0594	3.192913386	1.63	0.003837
DES Encrypted Source File							
CBC	7.984761	0	43.55	126.5590	3.187007874	1.45	-0.005805
CFB	7.985555	0	66.11	127.9566	3.055118110	2.75	-0.001727
OFB	7.985568	0	69.94	127.4976	3.141732283	0.00	0.004748
Duplicate Bytes Removed File							
NONE	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
AES Encrypted Duplicate Bytes Removed File							
CBC	7.987448	0	98.47	127.0578	3.208459215	2.13	-0.001078
CFB	7.983033	0	12.66	128.0075	3.125881168	0.50	-0.006565
OFB	7.984573	0	49.83	126.9382	3.170191339	0.91	-0.000054
DES Encrypted Duplicate Bytes Removed File							
CBC	7.985959	0	84.14	128.4050	3.121410579	0.64	-0.001444
CFB	7.983799	0	31.09	126.4731	3.117380353	0.77	-0.012860
OFB	7.986016	0	86.87	127.3440	3.153652393	0.38	0.005629
Duplicate Blocks Removed File							
NONE	4.374020	45	0.01	93.6546	4.000000000	27.32	-0.038895
AES Encrypted Duplicate Blocks Removed File							
CBC	7.982608	0	15.99	129.0339	3.187795897	1.47	-0.005523
CFB	7.985064	0	79.47	127.8787	3.109942136	1.01	-0.003268
OFB	7.985196	0	84.88	127.3972	3.122567070	0.61	-0.001105
DES Encrypted Duplicate Blocks Removed File							
CBC	7.985360	0	85.94	126.2659	3.157894737	0.52	-0.008137
CFB	7.982293	0	15.19	127.9438	3.086315789	1.76	-0.007784
OFB	7.983706	0	48.89	127.4754	3.164210526	0.72	-0.005406

Table 5.3 Results of ENT tests on Image File

IMAGE FILE							
Encryption type/ mode	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	7.184481	10	0.01	126.8271	3.504828474	11.56	0.435878
AES Encrypted Source File							
CBC	7.997009	0	25.02	127.6634	3.159095089	0.56	0.002612
CFB	7.997205	0	53.19	127.9309	3.133345595	0.26	0.002743
OFB	7.997005	0	21.39	127.6488	3.139231194	0.08	-0.005302
DES Encrypted Source File							
CBC	7.997578	0	94.72	127.4071	3.165348538	0.76	0.001454
CFB	7.997158	0	46.28	127.1562	3.133713445	0.25	0.005772
OFB	7.997188	0	48.69	127.5111	3.131506345	0.32	-0.000608
Duplicate Bytes Removed File							
NONE	7.203551	9	0.01	127.0198	3.497174040	11.32	0.403755
AES Encrypted Duplicate Bytes Removed File							
CBC	7.997269	0	82.82	127.1143	3.156664069	0.48	-0.005372
CFB	7.997007	0	48.93	127.3832	3.131722525	0.31	-0.002387
OFB	7.996878	0	29.65	127.9351	3.147310990	0.18	0.000137
DES Encrypted Duplicate Bytes Removed File							
CBC	7.996863	0	27.32	127.3771	3.149873319	0.26	-0.004107
CFB	7.997184	0	73.21	127.3175	3.160397583	0.60	0.005344
OFB	7.996711	0	11.82	127.2607	3.121808614	0.63	0.006084
Duplicate Blocks Removed File							
NONE	7.203451	9	0.01	127.0309	3.497125037	11.32	0.403595
AES Encrypted Duplicate Blocks Removed File							
CBC	7.997275	0	83.41	127.1037	3.156664069	0.48	-0.005419
CFB	7.997005	0	48.53	127.3841	3.131722525	0.31	-0.002413
OFB	7.996875	0	29.28	127.9343	3.147310990	0.18	0.000151
DES Encrypted Duplicate Blocks Removed File							
CBC	7.996865	0	27.67	127.3809	3.149873319	0.26	-0.004063
CFB	7.997172	0	71.66	127.3166	3.160397583	0.60	0.005361
OFB	7.996707	0	11.58	127.2616	3.121808614	0.63	0.006112

Table 5.4 Results of ENT tests on Audio File

AUDIO FILE							
Encryption type/ mode	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	6.200233	22	0.01	125.2086	2.295751129	26.92	0.174946
AES Encrypted Source File							
CBC	7.998017	0	29.11	127.9112	3.109956710	1.01	0.007598

CFB	7.997902	0	11.73	127.3821	3.130488559	0.35	0.007982
OFB	7.998031	0	32.37	127.9757	3.119851577	0.69	-0.001320
DES Encrypted Source File							
CBC	7.997967	0	21.63	127.5902	3.126283241	0.49	0.005749
CFB	7.998078	0	42.50	127.4880	3.160667904	0.61	-0.004354
OFB	7.997968	0	20.95	127.2620	3.147062461	0.17	-0.001750
Duplicate Bytes Removed File							
NONE	6.238276	22	0.01	125.9610	2.267692885	27.82	0.160875
AES Encrypted Duplicate Bytes Removed File							
CBC	7.998085	0	51.59	127.4865	3.140890891	0.02	-0.004667
CFB	7.998121	0	57.31	127.6669	3.136636637	0.16	0.001433
OFB	7.998058	0	43.98	127.3267	3.131131131	0.33	-0.004011
DES Encrypted Duplicate Bytes Removed File							
CBC	7.998247	0	84.00	127.6452	3.145789013	0.13	0.001662
CFB	7.998192	0	73.09	127.4565	3.165310975	0.75	-0.000014
OFB	7.998354	0	94.66	127.5859	3.148291828	0.21	-0.002491
Duplicate Blocks Removed File							
NONE	6.238962	22	0.01	125.9726	2.267651477	27.82	0.160845
AES Encrypted Duplicate Blocks Removed File							
CBC	7.998095	0	53.99	127.4925	3.142320691	0.02	-0.004222
CFB	7.998103	0	53.95	127.6487	3.138815872	0.09	0.000346
OFB	7.998045	0	41.08	127.3267	3.129302791	0.39	-0.004233
DES Encrypted Duplicate Blocks Removed File							
CBC	7.998241	0	83.44	127.6534	3.145772047	0.13	0.001940
CFB	7.998220	0	78.57	127.4175	3.163547600	0.70	-0.000599
OFB	7.998339	0	93.56	127.6059	3.147524567	0.19	-0.002567

Table 5.5 Results of ENT tests on Video File

VIDEO FILE							
Encryption type/ mode	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	7.606804	4	0.01	112.1045	3.271680939	4.14	0.256775
AES Encrypted Source File							
CBC	7.999953	0	21.07	127.4872	3.141693605	0.00	0.000406
CFB	7.999959	0	78.20	127.4771	3.144166945	0.08	0.000367
OFB	7.999956	0	47.08	127.4854	3.139824220	0.06	-0.000285
DES Encrypted Source File							
CBC	7.999956	0	49.38	127.4710	3.138742853	0.09	0.000121
CFB	7.999956	0	45.43	127.4665	3.141716613	0.00	0.000472
OFB	7.999957	0	57.75	127.5061	3.142228537	0.02	0.000206
Duplicate Bytes Removed File							
NONE	7.892749	1	0.01	112.4897	3.389071855	7.88	0.031485
AES Encrypted Duplicate Bytes Removed File							
CBC	7.999945	0	10.11	127.5300	3.141478682	0.00	-0.000809
CFB	7.999952	0	62.23	127.5222	3.136860907	0.15	-0.000506
OFB	7.999961	0	99.18	127.4717	3.143813224	0.07	-0.000152
DES Encrypted Duplicate Bytes Removed File							

CBC	7.999953	0	66.48	127.4791	3.142227695	0.02	0.001001
CFB	7.999948	0	21.64	127.4859	3.142317485	0.02	0.000002
OFB	7.999951	0	54.55	127.5118	3.142253349	0.02	-0.000318
Duplicate Blocks Removed File							
NONE	7.897290	1	0.01	112.7841	3.388054407	7.85	0.030452
AES Encrypted Duplicate Blocks Removed File							
CBC	7.999955	0	83.72	127.5514	3.139340839	0.07	0.000570
CFB	7.999952	0	60.29	127.4694	3.143507150	0.06	-0.000383
OFB	7.999948	0	30.82	127.4848	3.143204028	0.05	-0.000172
DES Encrypted Duplicate Blocks Removed File							
CBC	7.999954	0	74.81	127.5190	3.139901937	0.05	0.000144
CFB	7.999952	0	59.27	127.4979	3.142462348	0.03	-0.000867
OFB	7.999945	0	8.83	127.5065	3.140256654	0.04	-0.000513

5.3.2 NIST Tests

Table 5.6 Results of NIST tests on Text – Source File

Text – Source File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	S	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 5.7 Results of NIST tests on Text – Duplicate Bytes Removed File

Text – Duplicate Bytes Removed File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	S	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S

Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	F	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	F	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 5.8 Results of NIST tests on Text – Duplicate Blocks Removed File

Text – Duplicate Blocks Removed File							
TEST	**UNENCRYPTED**	**AES ENCRYPTED**			**DES ENCRYPTED**		
		CBC	**CFB**	**OFB**	**CBC**	**CFB**	**OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	S	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 5.9 Results of NIST tests on Image – Source File

Image – Source File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 5.10 Results of NIST tests on Image – Duplicate Bytes Removed File

Image – Duplicate Bytes Removed File							
TEST	UNENCRYPTED	ENCRYPTED					
		DES CBC	DES CFB	DES OFB	AES CBC	AES CFB	AES OFB
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	F	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 5.11 Results of NIST tests on Image – Duplicate Blocks Removed File

Image – Duplicate Blocks Removed File							
TEST	UNENCRYPTED	AES ENCRYPTED			DES ENCRYPTED		
		CBC	CFB	OFB	CBC	CFB	OFB
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	F	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 5.12 Results of NIST tests on Audio – Source File

Audio – Source File							
TEST	UNENCRYPTED	ENCRYPTED					
		DES CBC	DES CFB	DES OFB	AES CBC	AES CFB	AES OFB
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S

Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	F	S	S	S	S
Serial1	F	S	F	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 5.13 Results of NIST tests on Audio – Duplicate bytes Removed File

Audio – Duplicate bytes Removed File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	F	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 5.14 Results of NIST tests on Audio – Duplicate Blocks Removed File

Audio – Duplicate Blocks Removed File							
TEST	**UNENCRYPTED**	**AES ENCRYPTED**			**DES ENCRYPTED**		
		CBC	**CFB**	**OFB**	**CBC**	**CFB**	**OFB**
Approx. Entropy	F	**S**	**S**	**S**	**S**	**S**	**S**
Block Frequency	F	**S**	**S**	**S**	**S**	**S**	**S**
Cumulative Sum	F	**S**	**S**	**S**	**S**	**S**	**S**
FFT	F	**S**	**S**	**S**	**S**	**S**	**S**
Frequency	F	**S**	**S**	**S**	**S**	**S**	**S**
Linear Complexity	S	**S**	**S**	**S**	**S**	**S**	**S**
Longest Run	S	**S**	**S**	**S**	**S**	**S**	**S**
Non-periodic template	S	**S**	**S**	**S**	**S**	**S**	**S**
Overlapping Template	F	**S**	**S**	**S**	**S**	**S**	**S**
Rank	F	**S**	**S**	**S**	**S**	**S**	**S**
Runs	F	**S**	**S**	**S**	**F**	**S**	**S**
Serial1	F	**S**	**S**	**S**	**S**	**S**	**S**
Serial2	F	**S**	**S**	**S**	**S**	**S**	**S**
Universal	S	**S**	**S**	**S**	**S**	**S**	**S**

Table 5.15 Results of NIST tests on Video – Source File

Video – Source File							
TEST	UNENCRYPTED	ENCRYPTED					
		DES CBC	DES CFB	DES OFB	AES CBC	AES CFB	AES OFB
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	F	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	F	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 5.16 Results of NIST tests on Video – Duplicate Bytes Removed File

Video – Duplicate Bytes Removed File							
TEST	UNENCRYPTED	ENCRYPTED					
		DES CBC	DES CFB	DES OFB	AES CBC	AES CFB	AES OFB
Approx. Entropy	F	S	S	F	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	F	S	S	S
Universal	S	S	S	S	S	S	S

Table 5.17 Results of NIST tests on Video – Duplicate Blocks Removed File

Video – Duplicate Blocks Removed File							
TEST	UNENCRYPTED	AES ENCRYPTED			DES ENCRYPTED		
		CBC	CFB	OFB	CBC	CFB	OFB
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S

Longest Run	S	**S**	**S**	**S**	**S**	**S**	**S**
Non-periodic template	S	**S**	**S**	**S**	**S**	**S**	**S**
Overlapping Template	F	**S**	**S**	**S**	**S**	**S**	**S**
Rank	F	**S**	**S**	**S**	**S**	**S**	**S**
Runs	F	**S**	**S**	**S**	**S**	**S**	**S**
Serial1	F	**S**	**S**	**S**	**S**	**S**	**S**
Serial2	F	**S**	**S**	**S**	**S**	**S**	**S**
Universal	S	**S**	**S**	**S**	**S**	**S**	**S**

5.4 Advantages and Limitations

The algorithm mainly reduces the problem of large size key distribution to the authorized receiver. One just requires to tell the authorized receiver (in a secure way), the song or movie name or link (for example) with which the data file is encrypted, the encryption algorithm the initial value and the secret key. The authorized receiver can generate the key file at his own site using the same popular song or movie file and decrypt the data file from it. An unauthorized receiver (intruder) does not have the knowledge of which song or movie file is used as key or even the extension of the file used for key generation and hence cannot generate key file. Even if the unauthorized receiver knows the source file from which the key file is generated, he cannot generate the key file as he does not know the secret key and initial value used at the encryption stage. Moreover the brute force attack on source file using the file database is not possible as the file database on the Internet is infinite. Also a brute force attack on the source file bits is not possible as the size of the source file is very large.

This algorithm is better than the algorithm given in chapter 4 in that it uses an initial vector and chaining mode encryption methods and hence the strength of the algorithm against the brute force attack is higher.

Again, this algorithm is slow.

Also, the source files downloaded by the authorized sender and receiver must be identical. To ensure this, the authorized sender and receiver systems can be synchronized.

The seed information, i.e. the source file link, the secret key and the initial value, must be transmitted to an authorized receiver in a secure way and thus rely on some other encryption techniques like RSA or AES. Hence the pseudorandom key based stream ciphers do not work completely independent of other algorithms and their security may be affected by any attack on the security of these algorithms.

CHAPTER 6

WITHOUT DUPLICATE BLOCKS REMOVAL-CHAINING MODE ENCRYPTION METHOD

6.1 Introduction

This is the fifth proposed algorithm and is an encryption based algorithm. The authorized sender and receiver can share their very large size data over an unsecure channel at a very low cost using this algorithm. For this, the authorized sender and the authorized receiver share a link on the Internet from where a file can be downloaded. This link must be shared using a very secure form of communication so that an unauthorized receiver is not able to see it. The authorized sender and receiver also decide the cryptographic symmetric key algorithm to be used during the process and share the secret key and the initial value between them through the same secure channel. Then they download the file from the Internet using this link, which is termed as the source file. Then both the authorized sender and receiver generate the key file using this source file. This key file is XORed with the plain data file to generate a cipher data file. Then this cipher data file is transmitted over the unsecure channel.

The information that goes from the authorized sender to the authorized receiver through the secure channel or medium includes: the file download link, the encryption algorithm, the chaining mode, the initial vector and the secret key

The information that goes from the authorized sender to the authorized receiver through the insecure channel or medium includes: the cipher data file encrypted using the key file generated from the proposed method.

6.2 Methodology

The key generation algorithm consists of the following steps.

Step 1: Input the source binary file in binary read mode. Open two other binary files, say X and Y in binary write mode.

Step 2: Remove the header from the source file. Store the result in X. Also, instead of generating the key from the whole file, we can also generate the key from a segment of the source file. This will be useful when the size of key is required to be very small as compared to the size of source file. This will speed up the process.

Step 3: Close X.

Step 4: Open X in binary read mode.

Step 5: Segment X in 64-bit blocks for DES and IDEA or 128-bit blocks for AES.

Step 6: Encrypt X using some standard secret key encryption algorithm like IDEA, DES or AES in Cyclic mode (CBC, CFB or OFB). Remove all duplicate bytes which occur eight times in a sequence blocks from the file. This increases randomness of the file. Also the secret key and the initial vector are only available with the authorized sender and receiver. Hence only the authorized sender and receiver can generate the file.

Step 7: Store the resultant bytes in Y. The resultant file Y serves as the key file during encryption.

Step 8: Close all files.

The significance of the major steps of the algorithm is given as follows:

Header Removal: The header is removed as the header of all files with same extension will be almost similar. Thus, if not removed, a cryptanalyst can easily predict the first few bytes of the key sequence. Also, instead of generating the key from the whole file, we can also generate the key from a segment of the source file. This will be useful when the size of key is required to be very small as compared to the size of source file. This will speed up the process.

Encryption: The results of NIST and ENT tests show that after encryption the randomness of the file increases to a great extent. The randomness of an encrypted file is close to the randomness of a true random file. The encryption is performed in chaining mode (CBC/CFB/OFB) [56] and not in ECB [56] mode because in ECB mode, a codebook attack is possible. In this attack, if two or more blocks of plaintext are identical then the corresponding ciphertext blocks will also be identical. Thus a cryptanalyst can use this information to attack the secret system. It does not matter that the identical blocks of the plaintext reside at which position in the source file. The probability of a codebook attack increases when the number of such identical blocks is large in the source file. Performing the encryption in chaining mode removes the probability of the codebook attack.

6.3 Results

The proposed algorithm was implemented in C and Java programming language to generate the pseudorandom binary key files from various types of source files (i.e. from the text, audio, video and image files). The change in the size of the files after each step is given in table 6.1. We also performed ENT [31] and NIST [32] statistical tests on the source files and the corresponding output files of the key generation algorithm to find the amount of randomness in the files. The ENT and the NIST tests are performed using the standard testing software available online on the official websites of these tests. The results of the ENT tests are given in the tables 6.2, 6.3, 6.4 and 6.5 for text, image, audio and video files respectively. The results of NIST tests for the text file are given in tables 6.6 and 6.7. The results of NIST tests for the image file are given in tables 6.8 and 6.9. The results of NIST tests for the audio file are given in tables 6.10 and 6.11. The results of NIST tests for the video file are given in tables 6.12 and 6.13.

Table 6.1 Size of the files after each step

File Type	Size of SF	Size of DRF	Size of KF
Text	11.8 KB	11.6 KB	11.6 KB
Audio	94.7 KB	93.6 KB	93.6 KB

Video	3.97 MB	3.56 MB	3.56 MB
Image	63.7 KB	60.1 KB	60.1 KB

6.3.1 ENT Tests

Table 6.2 Results of ENT tests on Text File

TEXT FILE							
Encryption type/ mode	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	4.364403	45	0.01	93.5753	4.000000000	27.32	0.005331
AES Encrypted Source File							
CBC	7.981414	0	0.85	129.2305	3.059055118	2.63	0.003301
CFB	7.984934	0	48.98	128.0091	3.088582677	1.69	-0.012361
OFB	7.984554	0	35.51	127.0594	3.192913386	1.63	0.003837
DES Encrypted Source File							
CBC	7.984761	0	43.55	126.5590	3.187007874	1.45	-0.005805
CFB	7.985555	0	66.11	127.9566	3.055118110	2.75	-0.001727
OFB	7.985568	0	69.94	127.4976	3.141732283	0.00	0.004748
Duplicate Bytes Removed File							
NONE	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
AES Encrypted Duplicate Bytes Removed File							
CBC	7.987448	0	98.47	127.0578	3.208459215	2.13	-0.001078
CFB	7.983033	0	12.66	128.0075	3.125881168	0.50	-0.006565
OFB	7.984573	0	49.83	126.9382	3.170191339	0.91	-0.000054
DES Encrypted Duplicate Bytes Removed File							
CBC	7.985959	0	84.14	128.4050	3.121410579	0.64	-0.001444
CFB	7.983799	0	31.09	126.4731	3.117380353	0.77	-0.012860
OFB	7.986016	0	86.87	127.3440	3.153652393	0.38	0.005629

Table 6.3 Results of ENT tests on Image File

IMAGE FILE							
Encryption type/ mode	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	7.184481	10	0.01	126.8271	3.504828474	11.56	0.435878
AES Encrypted Source File							
CBC	7.997009	0	25.02	127.6634	3.159095089	0.56	0.002612
CFB	7.997205	0	53.19	127.9309	3.133345595	0.26	0.002743
OFB	7.997005	0	21.39	127.6488	3.139231194	0.08	-0.005302
DES Encrypted Source File							
CBC	7.997578	0	94.72	127.4071	3.165348538	0.76	0.001454
CFB	7.997158	0	46.28	127.1562	3.133713445	0.25	0.005772

OFB	7.997188	0	48.69	127.5111	3.131506345	0.32	-0.000608
Duplicate Bytes Removed File							
NONE	7.203551	9	0.01	127.0198	3.497174040	11.32	0.403755
AES Encrypted Duplicate Bytes Removed File							
CBC	7.997269	0	82.82	127.1143	3.156664069	0.48	-0.005372
CFB	7.997007	0	48.93	127.3832	3.131722525	0.31	-0.002387
OFB	7.996878	0	29.65	127.9351	3.147310990	0.18	0.000137
DES Encrypted Duplicate Bytes Removed File							
CBC	7.996863	0	27.32	127.3771	3.149873319	0.26	-0.004107
CFB	7.997184	0	73.21	127.3175	3.160397583	0.60	0.005344
OFB	7.996711	0	11.82	127.2607	3.121808614	0.63	0.006084

Table 6.4 Results of ENT tests on Audio File

AUDIO FILE							
Encryption type/ mode	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	6.200233	22	0.01	125.2086	2.295751129	26.92	0.174946
AES Encrypted Source File							
CBC	7.998017	0	29.11	127.9112	3.109956710	1.01	0.007598
CFB	7.997902	0	11.73	127.3821	3.130488559	0.35	0.007982
OFB	7.998031	0	32.37	127.9757	3.119851577	0.69	-0.001320
DES Encrypted Source File							
CBC	7.997967	0	21.63	127.5902	3.126283241	0.49	0.005749
CFB	7.998078	0	42.50	127.4880	3.160667904	0.61	-0.004354
OFB	7.997968	0	20.95	127.2620	3.147062461	0.17	-0.001750
Duplicate Bytes Removed File							
NONE	6.238276	22	0.01	125.9610	2.267692885	27.82	0.160875
AES Encrypted Duplicate Bytes Removed File							
CBC	7.998085	0	51.59	127.4865	3.140890891	0.02	-0.004667
CFB	7.998121	0	57.31	127.6669	3.136636637	0.16	0.001433
OFB	7.998058	0	43.98	127.3267	3.131131131	0.33	-0.004011
DES Encrypted Duplicate Bytes Removed File							
CBC	7.998247	0	84.00	127.6452	3.145789013	0.13	0.001662
CFB	7.998192	0	73.09	127.4565	3.165310975	0.75	-0.000014
OFB	7.998354	0	94.66	127.5859	3.148291828	0.21	-0.002491

Table 6.5 Results of ENT tests on Video File

VIDEO FILE							
Encryption type/ mode	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	7.606804	4	0.01	112.1045	3.271680939	4.14	0.256775

AES Encrypted Source File							
CBC	7.999953	0	21.07	127.4872	3.141693605	0.00	0.000406
CFB	7.999959	0	78.20	127.4771	3.144166945	0.08	0.000367
OFB	7.999956	0	47.08	127.4854	3.139824220	0.06	-0.000285
DES Encrypted Source File							
CBC	7.999956	0	49.38	127.4710	3.138742853	0.09	0.000121
CFB	7.999956	0	45.43	127.4665	3.141716613	0.00	0.000472
OFB	7.999957	0	57.75	127.5061	3.142228537	0.02	0.000206
Duplicate Bytes Removed File							
NONE	7.892749	1	0.01	112.4897	3.389071855	7.88	0.031485
AES Encrypted Duplicate Bytes Removed File							
CBC	7.999945	0	10.11	127.5300	3.141478682	0.00	-0.000809
CFB	7.999952	0	62.23	127.5222	3.136860907	0.15	-0.000506
OFB	7.999961	0	99.18	127.4717	3.143813224	0.07	-0.000152
DES Encrypted Duplicate Bytes Removed File							
CBC	7.999953	0	66.48	127.4791	3.142227695	0.02	0.001001
CFB	7.999948	0	21.64	127.4859	3.142317485	0.02	0.000002
OFB	7.999951	0	54.55	127.5118	3.142253349	0.02	-0.000318

6.3.2 NIST Tests

Table 6.6 Results of NIST tests on Text – Source File

Text – Source File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	S	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 6.7 Results of NIST tests on Text – Duplicate Removed File

Text – Duplicate Removed File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	S	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S

Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	F	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	F	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 6.8 Results of NIST tests on Image – Source File

Image – Source File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 6.9 Results of NIST tests on Image – Duplicate Removed File

Image – Duplicate Removed File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	F	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 6.10 Results of NIST tests on Audio – Source File

Audio – Source File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	F	S	S	S	S
Serial1	F	S	F	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 6.11 Results of NIST tests on Audio – Duplicate Removed File

Audio – Duplicate Removed File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	F	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 6.12 Results of NIST tests on Video – Source File

Video – Source File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S

Linear Complexity	F	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	F	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 6.13 Results of NIST tests on Video – Duplicate Removed File

Video – Duplicate Removed File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	F	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	F	S	S	S
Universal	S	S	S	S	S	S	S

6.4 Advantages and Limitations

The algorithm mainly reduces the problem of large size key distribution to the authorized receiver. One just requires to tell the authorized receiver (in a secure way), the song or movie name or link (for example) with which the data file is encrypted, the encryption algorithm the initial value and the secret key. The authorized receiver can generate the key file at his own site using the same popular song or movie file and decrypt the data file from it. An unauthorized receiver (intruder) does not have the knowledge of which song or movie file is used as key or even the extension of the file used for key generation and hence cannot generate key file. Even if the unauthorized receiver knows the source file from which the key file is generated, he cannot generate the key file as he does not know the secret key and initial value used at the encryption stage. Moreover the brute force attack on source file using the file database is not possible as the file database on the Internet is infinite. Also a brute force attack on the source file bits is not possible as the size of the source file is very large.

This algorithm is faster than the algorithms given in chapter 4 and 5. It uses initial vector as an extra security weapon. But this algorithm is somewhat less (almost negligible) secure.

Also, the source files downloaded by the authorized sender and receiver must be identical. To ensure this, the authorized sender and receiver systems can be synchronized.

The seed information, i.e. the source file link, the secret key and the initial value, must be transmitted to an authorized receiver in a secure way and thus rely on some other encryption techniques like RSA or AES. Hence the pseudorandom key based stream ciphers do not work completely independent of other algorithms and their security may be affected by any attack on the security of these algorithms.

CHAPTER 7

WITHOUT DUPLICATE BLOCKS REMOVAL- NULL REMOVED- CHAINING MODE ENCRYPTION METHOD

7.1 Introduction

This is the sixth proposed algorithm and is an encryption based algorithm. The authorized sender and receiver can share their very large size data over an unsecure channel at a very low cost using this algorithm. For this, the authorized sender and the authorized receiver share a link on the Internet from where a file can be downloaded. This link must be shared using a very secure form of communication so that an unauthorized receiver is not able to see it. The authorized sender and receiver also decide the cryptographic symmetric key algorithm to be used during the process and share the secret key and the initial vector between them through the same secure channel. Then they download the file from the Internet using this link, which is termed as the source file. Then both the authorized sender and receiver generate the key file using this source file. This key file is XORed with the plain data file to generate a cipher data file. Then this cipher data file is transmitted over the unsecure channel.

The information that goes from the authorized sender to the authorized receiver through the secure channel or medium includes: the file download link, the encryption algorithm, chaining mode, the initial vector and the secret key.

The information that goes from the authorized sender to the authorized receiver through the insecure channel or medium includes: the cipher data file encrypted using the key file generated from the proposed method.

7.2 Methodology

The key generation algorithm consists of the following steps.

Step 1: Input the source binary file in binary read mode. Open two other binary files, say X and Y in binary write mode.

Step 2: Remove the header from the source file. Store the result in X. Also, instead of generating the key from the whole file, we can also generate the key from a segment of the source file. This will be useful when the size of key is required to be very small as compared to the size of source file. This will speed up the process.

Step 3: Close X.

Step 4: Open X in binary read mode.

Step 5: Segment X in 64-bit blocks for DES and IDEA or 128-bit blocks for AES.

Step 6: Encrypt X using some standard secret key encryption algorithm like IDEA, DES or AES in Cyclic mode (CBC, CFB or OFB). Remove all null bytes which occur eight times in a sequence blocks from the file or alternatively remove all duplicate bytes which occur eight times in a sequence blocks from the file. This increases randomness of the file. Also the secret key and the initial vector are only available with the authorized sender and receiver. Hence only the authorized sender and receiver can generate the file.

Step 7: Store the resultant bytes in Y. The resultant file Y serves as the key file during encryption.

Step 8: Close all files.

The significance of the major steps of the algorithm is given as follows:

Header Removal: The header is removed as the header of all files with same extension will be almost similar. Thus, if not removed, a cryptanalyst can easily predict the first few bytes of the key sequence. Also, instead of generating the key from the whole file, we can also generate the key from a segment of the source file. This will be useful when the size of key is required to be very small as compared to the size of source file. This will speed up the process.

Encryption: The results of NIST [32] and ENT [31] tests show that after encryption the randomness of the file increases to a great extent. The randomness of an encrypted file is close to the randomness of a true random file. The encryption is performed in chaining mode (CBC/CFB/OFB) [56] and not in ECB [56] mode because in ECB mode, a codebook attack is possible. In this attack, if two or more blocks of plaintext are identical then the corresponding ciphertext blocks will also be identical. Thus a cryptanalyst can use this information to attack the secret system. It does not matter that the identical blocks of the plaintext reside at which position in the source file. The probability of a codebook attack increases when the number of such identical blocks is large in the source file. Performing the encryption in chaining mode removes the probability of the codebook attack.

7.3 Results

The proposed algorithm was implemented in C and Java programming language to generate the pseudorandom binary key files from various types of source files (i.e. from the text, audio, video and image files). The change in the size of the files after each step is given in table 7.1. We also performed ENT [31] and NIST [31] statistical tests on the source files and the corresponding output files of the key generation algorithm to find the amount of randomness in the files. The ENT and the NIST tests are performed using the standard testing software available online on the official websites of these tests. The results of the ENT tests are given in the tables 7.2, 7.3, 7.4 and 7.5 for text, image, audio and video files respectively. The results of NIST tests for the text file are given in tables 7.6, 7.7 and 7.8. The results of NIST tests for the image file are given in tables 7.9, 7.10 and 7.11. The results of NIST tests for the audio file are given in tables 7.12, 7.13 and 7.14. The results of NIST tests for the video file are given in tables 7.15, 7.16 and 7.17.

Table 7.1 Size of the files after each step

File Type	Size of SF	Size of NRF	Size of DRF	Size of KF
Text	11.8 KB	11.8 KB	11.6 KB	11.6 KB
Audio	94.7 KB	61.3 KB	61 KB	61 KB
Video	3.97 MB	3.5 MB	3.47 MB	3.47 MB
Image	63.7 KB	61.5 KB	58 KB	58 KB

7.3.1 ENT Tests

Table 7.2 Results of ENT tests on Text File

TEXT FILE							
Encryption type/ mode	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	4.364403	45	0.01	93.5753	4.000000000	27.32	0.005331
Source file encrypted by DES in different chaining modes							
CBC	7.984761	0	43.55	126.5590	3.187007874	1.45	-0.005805
CFB	7.985555	0	66.11	127.9566	3.055118110	2.75	-0.001727
OFB	7.985568	0	69.94	127.4976	3.141732283	0.00	0.004748
Source file encrypted by AES in different chaining modes							
CBC	7.981414	0	0.85	129.2305	3.059055118	2.63	0.003301
CFB	7.984934	0	48.98	128.0091	3.088582677	1.69	-0.012361
OFB	7.984554	0	35.51	127.0594	3.192913386	1.63	0.003837
Bytes with ASCII value 0 or 255 removed File							
NONE	4.364403	45	0.01	93.5753	4.000000000	27.32	0.005331
Bytes with ASCII value 0 or 255 removed file encrypted by DES in different chaining modes							
CBC	7.984761	0	43.55	126.5590	3.187007874	1.45	-0.005805
CFB	7.985555	0	66.11	127.9566	3.055118110	2.75	-0.001727
OFB	7.985568	0	69.94	127.4976	3.141732283	0.00	0.004748
Bytes with ASCII value 0 or 255 removed file encrypted by AES in different chaining modes							
CBC	7.981414	0	0.85	129.2305	3.059055118	2.63	0.003301
CFB	7.984934	0	48.98	128.0091	3.088582677	1.69	-0.012361
OFB	7.984554	0	35.51	127.0594	3.192913386	1.63	0.003837
Duplicate bytes removed File							
NONE	4.371993	45	0.01	93.8352	4.000000000	27.32	-0.034203
Duplicate bytes removed file encrypted by DES in different chaining modes							
CBC	7.985959	0	84.14	128.4050	3.121410579	0.64	-0.001444
CFB	7.983799	0	31.09	126.4731	3.117380353	0.77	-0.012860
OFB	7.986016	0	86.87	127.3440	3.153652393	0.38	0.005629
Duplicate bytes removed file encrypted by AES in different chaining modes							
CBC	7.987448	0	98.47	127.0578	3.208459215	2.13	-0.001078
CFB	7.983033	0	12.66	128.0075	3.125881168	0.50	-0.006565
OFB	7.984573	0	49.83	126.9382	3.170191339	0.91	-0.000054

Table 7.3 Results of ENT tests on Image File

IMAGE FILE							
Encryption type/ mode	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	7.184481	10	0.01	126.8271	3.504828474	11.56	0.435878
Source file encrypted by DES in different chaining modes							
CBC	7.997578	0	94.72	127.4071	3.165348538	0.76	0.001454
CFB	7.997158	0	46.28	127.1562	3.133713445	0.25	0.005772
OFB	7.997188	0	48.69	127.5111	3.131506345	0.32	-0.000608
Source file encrypted by AES in different chaining modes							
CBC	7.997009	0	25.02	127.6634	3.159095089	0.56	0.002612
CFB	7.997205	0	53.19	127.9309	3.133345595	0.26	0.002743
OFB	7.997005	0	21.39	127.6488	3.139231194	0.08	-0.005302
Bytes with ASCII value 0 or 255 removed File							
NONE	7.190326	10	0.01	124.0478	3.596536302	14.48	0.573973
Bytes with ASCII value 0 or 255 removed file encrypted by DES in different chaining modes							
CBC	7.997416	0	90.94	126.9423	3.148539347	0.22	-0.003961
CFB	7.997185	0	64.29	127.4188	3.164906271	0.74	0.001327
OFB	7.996693	0	6.61	127.7179	3.124559901	0.54	-0.000463
Bytes with ASCII value 0 or 255 removed file encrypted by AES in different chaining modes							
CBC	7.997119	0	54.02	127.5154	3.160338757	0.60	0.001292
CFB	7.997150	0	57.29	127.6159	3.129508041	0.38	-0.003875
OFB	7.996935	0	30.76	127.8547	3.140546198	0.03	0.003660
Duplicate bytes removed File							
NONE	7.213525	9	0.01	123.9359	3.612877182	15.00	0.548074
Duplicate bytes removed file encrypted by DES in different chaining modes							
CBC	7.997007	0	65.58	127.4817	3.156407669	0.47	-0.010896
CFB	7.997070	0	72.20	127.6587	3.142280525	0.02	0.000993
OFB	7.996509	0	7.62	127.7783	3.115640767	0.83	0.003125
Duplicate bytes removed file encrypted by AES in different chaining modes							
CBC	7.997006	0	61.67	127.8429	3.153349475	0.37	0.001354
CFB	7.996739	0	27.57	127.6573	3.113801453	0.88	-0.002511
OFB	7.996965	0	57.82	127.4581	3.135996772	0.18	-0.002745

TABLE 7.4 Results of ENT tests on Audio File

AUDIO FILE							
Encryption type/ mode	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	6.200233	22	0.01	125.2086	2.295751129	26.92	0.174946
Source file encrypted by DES in different chaining modes							
CBC	7.997967	0	21.63	127.5902	3.126283241	0.49	0.005749

CFB	7.998078	0	42.50	127.4880	3.160667904	0.61	-0.004354
OFB	7.997968	0	20.95	127.2620	3.147062461	0.17	-0.001750
Source file encrypted by AES in different chaining modes							
CBC	7.998017	0	29.11	127.9112	3.109956710	1.01	0.007598
CFB	7.997902	0	11.73	127.3821	3.130488559	0.35	0.007982
OFB	7.998031	0	32.37	127.9757	3.119851577	0.69	-0.001320
Bytes with ASCII value 0 or 255 removed File							
NONE	7.580974	5	0.01	125.6436	2.861509074	8.92	0.026084
Bytes with ASCII value 0 or 255 removed file encrypted by DES in different chaining modes							
CBC	7.997276	0	78.44	127.6637	3.116714422	0.79	-0.000042
CFB	7.997133	0	57.58	127.3520	3.111747851	0.95	0.000487
OFB	7.996821	0	15.65	126.9218	3.154918816	0.42	0.003979
Bytes with ASCII value 0 or 255 removed file encrypted by AES in different chaining modes							
CBC	7.997055	0	44.50	127.2969	3.152406417	0.34	-0.009549
CFB	7.997608	0	98.53	127.1199	3.167303285	0.82	0.007595
OFB	7.997290	0	79.87	127.3885	3.122994652	0.59	-0.003575
Duplicate bytes removed File							
NONE	7.585443	5	0.01	125.6610	2.866961029	8.74	0.019331
Duplicate bytes removed file encrypted by DES in different chaining modes							
CBC	7.996774	0	13.79	127.3606	3.142637742	0.03	0.003703
CFB	7.997372	0	89.35	127.4517	3.136110578	0.17	-0.002129
OFB	7.997070	0	49.54	127.2234	3.149164907	0.24	0.006220
Duplicate bytes removed file encrypted by AES in different chaining modes							
CBC	7.997469	0	95.30	127.3135	3.154924170	0.42	0.003221
CFB	7.997385	0	89.62	127.6495	3.148397005	0.22	0.002925
OFB	7.997027	0	45.56	127.4090	3.138798234	0.09	0.001511

TABLE 7.5 Results of ENT tests on Video File

VIDEO FILE							
Encryption type/ mode	**Entropy (bits/byte)**	**Optimal Compression Reduction %**	**Chi square Distribution %**	**Arithmetic mean**	**Monte Carlo Value for Pi**	**Monte Carlo Error %**	**Serial Correlation Coefficient**
The Desired Totally Random File							
NONE	8	0	10 to 90	127.5	3.14	0.0	0.0
Source File							
NONE	7.606804	4	0.01	112.1045	3.271680939	4.14	0.256775
Source file encrypted by DES in different chaining modes							
CBC	7.999956	0	49.38	127.4710	3.138742853	0.09	0.000121
CFB	7.999956	0	45.43	127.4665	3.141716613	0.00	0.000472
OFB	7.999957	0	57.75	127.5061	3.142228537	0.02	0.000206
Source file encrypted by AES in different chaining modes							
CBC	7.999953	0	21.07	127.4872	3.141693605	0.00	0.000406
CFB	7.999959	0	78.20	127.4771	3.144166945	0.08	0.000367
OFB	7.999956	0	47.08	127.4854	3.139824220	0.06	-0.000285
Bytes with ASCII value 0 or 255 removed File							
NONE	7.905912	1	0.01	114.2856	3.391006983	7.94	0.034345
Bytes with ASCII value 0 or 255 removed file encrypted by DES in different chaining modes							
CBC	7.999943	0	6.14	127.4424	3.143817882	0.07	0.000206
CFB	7.999952	0	66.28	127.4448	3.144183345	0.08	-0.000742
OFB	7.999948	0	35.60	127.4761	3.139569374	0.06	-0.001084
Bytes with ASCII value 0 or 255 removed file encrypted by AES in different chaining modes							

CBC	7.999953	0	71.47	127.4964	3.143799700	0.07	-0.000334
CFB	7.999952	0	69.72	127.4711	3.143649600	0.07	-0.000336
OFB	7.999956	0	90.93	127.4584	3.143134037	0.05	0.000310
Duplicate bytes removed File							
NONE	7.908754	1	0.01	114.5160	3.386551745	7.80	0.024494
Duplicate bytes removed file encrypted by DES in different chaining modes							
CBC	7.999947	0	26.11	127.5158	3.143274320	0.05	0.000797
CFB	7.999952	0	70.04	127.5023	3.145519642	0.12	-0.000162
OFB	7.999951	0	61.72	127.4883	3.142359071	0.02	-0.000699
Duplicate bytes removed file encrypted by AES in different chaining modes							
CBC	7.999958	0	97.83	127.5090	3.140054487	0.05	-0.000339
CFB	7.999956	0	91.77	127.4891	3.142457838	0.03	0.000619
OFB	7.999956	0	92.17	127.6123	3.140475897	0.04	-0.000358

7.3.2 NIST Tests

Table 7.6 Results of NIST tests on Text – Source File

Text – Source File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	S	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 7.7 Results of NIST tests on Text – ASCII value 0 or 255 removed File

Text – ASCII value 0 or 255 removed File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	S	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S

Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 7.8 Results of NIST tests on Text – Duplicate bytes removed File

Text – Duplicate bytes removed File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	S	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	F	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	F	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 7.9 Results of NIST tests on Image – Source File

Image – Source File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 7.10 Results of NIST tests on Image – ASCII value 0 or 255 removed File

Image – ASCII value 0 or 255 removed File							
TEST	UNENCRYPTED	ENCRYPTED					
		DES CBC	DES CFB	DES OFB	AES CBC	AES CFB	AES OFB
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	F
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 7.11 Results of NIST tests on Image – Duplicate bytes removed File

Image – Duplicate bytes removed File							
TEST	UNENCRYPTED	ENCRYPTED					
		DES CBC	DES CFB	DES OFB	AES CBC	AES CFB	AES OFB
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	F	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 7.12 Results of NIST tests on Audio – Source File

Audio – Source File							
TEST	UNENCRYPTED	ENCRYPTED					
		DES CBC	DES CFB	DES OFB	AES CBC	AES CFB	AES OFB
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S

Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	F	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	F	S	S	S	S
Serial1	F	S	F	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 7.13 Results of NIST tests on Audio – ASCII value 0 or 255 removed File

Audio – ASCII value 0 or 255 removed File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	S	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S	S
Rank	S	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 7.14 Results of NIST tests on Audio – Duplicate bytes removed File

Audio – Duplicate bytes removed File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	S	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	F	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S	S
Rank	S	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	S	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 7.15 Results of NIST tests on Video – Source File

Video – Source File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	F	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	F	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 7.16 Results of NIST tests on Video – ASCII value 0 or 255 removed File

Video – ASCII value 0 or 255 removed File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	S
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	F	S
FFT	F	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S
Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S	S
Rank	F	S	S	S	S	S	S
Runs	F	S	F	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

Table 7.17 Results of NIST tests on Video – Duplicate bytes removed File

Video – Duplicate bytes removed File							
TEST	**UNENCRYPTED**	**ENCRYPTED**					
		DES CBC	**DES CFB**	**DES OFB**	**AES CBC**	**AES CFB**	**AES OFB**
Approx. Entropy	F	S	S	S	S	S	F
Block Frequency	F	S	S	S	S	S	S
Cumulative Sum	F	S	S	S	S	S	S
FFT	S	S	S	S	S	S	S
Frequency	F	S	S	S	S	S	S

Linear Complexity	S	S	S	S	S	S	S
Longest Run	S	S	S	S	S	S	S
Non-periodic template	S	S	S	S	S	S	S
Overlapping Template	S	S	S	S	S	S	S
Rank	S	S	S	S	S	S	S
Runs	F	S	S	S	S	S	S
Serial1	F	S	S	S	S	S	S
Serial2	F	S	S	S	S	S	S
Universal	S	S	S	S	S	S	S

7.4 Advantages and Limitations

The algorithm mainly reduces the problem of large size key distribution to the authorized receiver. One just requires to tell the authorized receiver (in a secure way), the song or movie name or link (for example) with which the data file is encrypted, the encryption algorithm the initial value and the secret key. The authorized receiver can generate the key file at his own site using the same popular song or movie file and decrypt the data file from it. An unauthorized receiver (intruder) does not have the knowledge of which song or movie file is used as key or even the extension of the file used for key generation and hence cannot generate key file. Even if the unauthorized receiver knows the source file from which the key file is generated, he cannot generate the key file as he does not know the secret key and initial value used at the encryption stage. Moreover the brute force attack on source file using the file database is not possible as the file database on the Internet is infinite. Also a brute force attack on the source file bits is not possible as the size of the source file is very large.

This algorithm is faster than the algorithms given in chapter 4 and 5. It uses initial vector as an extra security weapon. But this algorithm is somewhat less (almost negligible) secure.

Also, the source files downloaded by the authorized sender and receiver must be identical. To ensure this, the authorized sender and receiver systems can be synchronized.

The seed information, i.e. the source file link, the secret key and the initial value, must be transmitted to an authorized receiver in a secure way and thus rely on some other encryption techniques like RSA or AES. Hence the pseudorandom key based stream ciphers do not work completely independent of other algorithms and their security may be affected by any attack on the security of these algorithms.

CONCLUSION

The proposed algorithms may prove to be a practical implementation of One-Time Pad if the source file and the secret key is not compromised in any way and the reduced bits from the source file in the key file are random enough. The results of our experiments show that the key file generated from the proposed algorithm is almost truly random. The proposed algorithms are simple, robust and universally applicable. The authorized sender and receiver can exchange large data including audio and video files at very low price during each session. In other words, the proposed algorithms can be used to generate very large size session keys to be used with stream cipher algorithms at a very low price.

The proposed algorithms require that the source files downloaded by the authorized sender and receiver must be identical. To ensure this, the authorized sender and receiver systems can be synchronized.

The seed information, i.e. the source file link and the secret value/secret key/initial value, must be transmitted to an authorized receiver in a secure way and thus rely on some other encryption techniques like RSA or AES. Hence the pseudorandom key based stream ciphers do not work completely independent of other algorithms and their security may be affected by any attack on the security of these algorithms.

ABBREVIATIONS

Acronym	Meaning
AES	Advance Encryption Standard
CBC	Cyclic Block Chaining Encryption mode
CF	Compressed File
CFB	Cyclic Feedback Encryption mode
DBRF	The file after removing blocks which are duplicate to any of their predecessor
DES	Data Encryption Standard
DRF	The file obtained after removing bytes which are duplicate to their immediate predecessor
EA	Encryption Algorithm applied in the implementation of the proposed algorithm for testing purpose
F	Failure
IDEA	International Data Encryption Algorithm
IF	The file obtained after applying the inversion process
KF	Key File generated as output from the proposed algorithm
N	The secret value used at the inversion step
NRF	The file obtained after removing bytes with ASCII values 0 or 255
OFB	Output Feedback Encryption mode
OTP	One Time Pad
PRNG	Pseudorandom number generator
S	Success
SF	Source File used as input to the proposed algorithm
UCF	Uncompressed File

REFERENCES

[1] Gilbert S. Vernam, *U. S. Patent 1310719*, "Secret signaling system", 22 July 1919.

[2] G. S. Vernam, "Cipher printing telegraph systems for secret wire and radio telegraphic communications", *J. of the American Institute of Electrical Engineers*, vol.55, Feb. 1926, pp. 295-301.

[3] C. E. Shannon, "Communication theory of secrecy systems", Bell Systems Technical J., vol. 28, 1949, pp. 656-715.

[4] William Stallings, *Cryptography and Network Security Principles and Practices*, 4th ed., USA: Prentice Hall, 2005, pp. 34, 48-49, 189-194, 218-227.

[5] I. J. Kumar, *Cryptology*, New York: Aegean Park Press, 1997, pp. 22-24, 138-249.

[6] Bruce Schneier, *Applied Cryptography*, 2nd ed., New York: Wiley, 1996, ch.2.8, 9, 12, 16, 17.

[7] Douglas Stinson, *Cryptography: Theory and Practice*, London: CRC press, 1995, ch. 12.

[8] Charlie Kaufman, Radia Perlman, Mike Speciner, *Network Security,* 2nd ed., India: PHI, 2002, pp. 59-104.

[9] Donald E. Knuth, "The Art of Computer Programming", 3rd ed., vol. 2: Seminumerical Algorithms. Reading, MA: Addison-Wesley, 1998, pp. 1-184.

[10] Jan L. Harrington, *Network Security,* India: Elsevier, 1st ed., 2006, pp. 286-288.

[11] Brijendra Singh, *Network Security and Management*, India: PHI, 2007, 1st ed., pp. 40.

[12] Richard E. Smith, *Internet Cryptography,* 2nd ed., Addison Weslay.

[13] K. Zeng, C. H. Yang, D. Y. Wei and T. R. N. Rao, "Pseudorandom bit generators in stream cipher cryptography", *Computer*, Feb. 1991, pp. 8-16.

[14] R. N. Mutagi, "Pseudo noise sequences for engineers", *Electronics and Communication Engineering J.*, pp. 79-87, April 1996.

[15] M. J. B. Robshaw, "Stream Ciphers", *RSA Laboratories Technical Report TR-701*, July 1995.

[16] Palash Sarkar, "Pseudo-random functions and parallelizable modes of operations of a block cipher", *IEEE Trans. on Information Theory*, vol.56, no. 8, Aug. 2010, pp. 4025-4037.

[17] Howard M. Heys, "Analysis of the statistical cipher feedback mode of block ciphers", *IEEE Trans. on Computers*, vol. 52, no. 1, 2003, pp. 77-92.

[18] Yang Xiao, Hsiao-Hwa Chen, Xiaojiang Du and Mohsen Guizani, "Stream-based cipher feedback mode in wireless error channel", IEEE Trans. Wireless Communications, vol. 8, no. 2, Feb. 2009, pp. 622-626.

[19] Debrup Chakraborty and Palash Sarkar, "A general construction of tweakable block ciphers and different modes of operations, *IEEE Trans. on Information Theory*, vol.54, no. 5, May 2008, pp. 1991-2006.

[20] Kamel H. Rahouma, "A block cipher technique for security of data and computer networks", *Int. Conf. on Internet Workshop*, IWS 1999.

[21] Ibrahim F. Elashry, Osama S. Farag Allah, Alaa M. Abbas and S. L. Rabaie, "A new diffusion mechanism for data encryption in the ECB mode", *Int. conf. on Computer Engineering and Systems*, ICCES 2009.

[22] Jun Yang, Lan Gao and Youtao Zang, "Improving memory encryption performance in secure processors", IEEE Trans. on Computers, vol. 54, no. 5, May 2005, pp.630-640.

[23] Timothy E. Lindquist, Mohamed Diarra and Bruce R. Millard, "A java cryptography service provider implementing One-Time pad", *Proc. of 37th Hawaii Int. Conf. on System Sciences*, 2004.

[24] Zhihua Chen and Jin Xu, "One-Time-Pad encryption in the tile assembly model", *3rd Int. Conf. on Bio-Inspired Computing: Theories and Applications*, BICTA 2008.

[25] Natalie Kostinski, Konstantin Kravtsov, and Paul R. Prucnal, "Demonstration of all-optical OCDMA encryption and decryption system with variable two-code keying", *IEEE Photonics technology letters*, vol. 20, n. 24, Dec. 2008, pp.2045-2047.

[26] Donghua Xu, Chenghuai Lu and Adre Dos Santos, "Protecting web usage of credit cards using One-Time pad cookie encryption", *Proc. of 18th Annual Computer Security Applications Conf.*, 2002.

[27] Jiao Hongqiang, Tian Junfeng and Wang Baomin, "A study on One-Time Pad scheme based on Stern-Brocot Tree", *Int. Sym. On Computer Science and Computational Technology*, 2008.

[28] Vasin Suttichaya and Pattarasinee Bhattarakosol, "Chain rule protection over the Internet using PUGGAD algorithm", *Int. Conf. on Computer and Electrical Engineering*, 2008.

[29] Chi-Wu Huang, Che-Hao Chiang, Chien-Lun Yen, Yi-Cheng Chen, Kuo-Huang Chang and Chi-Jeng Chang, "The AES application in image using different operation modes", *5th IEEE Conf. on Industrial Electronics and Applications*, 2010.

[30] Yan Zhang, Chengqi Xu and Feng Wang, "A novel scheme for secure network coding using One-Time Pad", *Int. Conf. on Network Security, Wireless Communication and Trusted Computing*, 2009

[31] "ENT. A pseudorandom number sequence test program", [Online]. Available: http://www.fourmilab.ch/random.

[32] "NIST Suite. Random Number Generation and Testing", [Online]. Available: http://csrc.nist.gov/rng/.

[33] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray and San Vo, "A statistical test suite for random and pseudorandom number generators for cryptographic applications", *NIST special publication 800-22* (with revision dated May 15, 2001).

[34] Alireza Yavari, "A practical research on randomness of digits of binary expansion of irrational numbers", *Int. Conf. on Information, Communications and signal Processing 2009*, ICICS 2009.

[35] Giles Cotter, "Generation of pseudorandom numbers from microphone input", *Computing Devices,* University of Virginia, 2002.

[36] Edward J. Groth, "Generation of binary sequences with controllable complexity", *IEEE Trans. on Information Theory*, vol. 17, pp. 288-296, May 1971.

[37] The Win RAR archiver website. [Online]. Available: http://www.rarlab.com.

[38] The Wikipedia website. [Online]. Available: http://www.wikipedia.com.

[39] The howstuffworks website. [Online]. Available: http://www.howstuffworks.com.

[40] Terry Ritter. (1991). The efficient generation of cryptographic confusion sequences. *Cryptologia* [Online]. vol. 15(2), pp. 81-139. Available: http://www.ciphersbyritter.com/ARTS/CRNG2ART.HTM

[41] Dilip V. Sarwate and Michael B. Pursley, "Crosscorrelation properties of pseudorandom and related sequences", *Proc. of IEEE*, vol. 68, no. 5, pp. 593-619, May 1980.

[42] Ai Hui Tan and Keith R. Godfrey, "The generation of binary and near-binary pseudorandom signals: An Overview", *IEEE Trans. On Instrumentations and Measurements*, vol. 51, no. 4, pp. 583-588, August 2002.

[43] Yukiyaso Tsunoo, Teruo Saito, Hiroyasu Kubo and Tomoyasu Suzaki, "A distinguishing attack on a fast software-implemented RC-4 like stream cipher", *IEEE Trans. on Information Theory*, vol. 53, no. 9, pp. 3250-3255 September 2007.

[44] Tao Sang, Ruli Wang, Tixun Yang, "Generating binary Bernoulli sequences based on a class of even-symmetric chaotic maps", *IEEE Trans. on Communications*, vol. 49, no. 4, pp. 620-623, April 2001.

[45] Jong-Seon No and P. Vijay Kumar, "A new family of binary pseudorandom sequences having optimal periodic correlation properties and large linear span", *IEEE Trans. on Information Theory*, vol. 35, no. 2, pp. 371-379, March 1989.

[46] Jong-Seon No, Solomon W. Golomb, Guang Gong, Hwan-Keun Lee and Peter Gaal, "Binary pseudorandom sequences of period $2^n - 1$ with ideal

autocorrelation", *IEEE Trans. on Information Theory*, vol. 44, no. 2, pp. 814-817, March 1998.

[47] Jong-Seon No, Habong Chung and Min-Seon Yun, "Binary pseudorandom sequences of period 2^m - 1 with ideal autocorrelation generated by the polynomial $Z^d + (Z+1)^d$", *IEEE Trans. on Information Theory*, vol. 44, no. 3, pp. 1278-1282, May 1998.

[48] Mark Goresky and Andrew Klapper, "Arithmetic cross-correlations of feedback with carry shift register sequences", *IEEE Trans. on Information Theory*, vol. 43, no. 4, pp. 1342-1345, July 1997.

[49] Chaoping Xing and Kwok Yan Lam, "Sequences with almost perfect linear complexity profiles and curves over finite fields", *IEEE Trans. on Information Theory*, vol. 45, no. 4, pp. 1267-1270, May 1999.

[50] Francois Arnault, Thiery P. Berger, and Abdelkadar Necer, "Feedback with carry shift registers synthesis with the Euclidean algorithm", *IEEE Trans. on Information Theory*, vol. 50, no. 5, pp. 910-917, May 2004.

[51] Xiaohu Tang, Parampalli Udaya, Pingzhi Fan, "A new family of nonbinary sequences with three level correlation property and large linear span", *IEEE Trans. on Information Theory*, vol. 51, no. 8, pp. 2906-2914, August 2005.

[52] Nilanjan Mukherji, Janusz Rajski, Grzegorz Mrugalski, Artur Pogiel and Jerzy Tyszer, "Ring generator: an ultimate linear feedback shift register", *Computer*, vol. 44, no. 6, pp. 64-71, June 2011.

[53] Howard M. Heys, "An analysis of the statistical self-synchronization of stream ciphers", *IEEE INFOCOM*, 2001.

[54] *Data Encryption Standard (DES)*, FIPS Publication 46-3, National Institute of Standard and Technology, 1979.

[55] *Advanced Encryption Standard (DES)*, FIPS Publication 197, National Institute of Standard and Technology, 2001.

[56] *DES modes of Operation*, FIPS Publication 81, National Institute of Standard and Technology, Dec. 1980.

[57] Recommendations for Block cipher modes of operation: Three variants of ciphertext stealing for CBC mode, FIPS Special Publication 800-38A, National Institute of Standard and Technology, Oct. 2010.