Crypto-Compression System: An Integrated Approach using Stream Cipher Cryptography and Entropy Encoding

Bobby Jasuja
Department of Information Technology
Medicaps Institute of Science and Technology,
Indore, India

Abhishek Pandya
Department of Information Technology
Medicaps Institute of Science and Technology,
Indore, India

ABSTRACT
Problems faced in modern communications are not only just related to security but also concerned with the communication speed and content size.Now day's networks demand exchange of information with more security and reduction in both -data storage and the time for data transmission. This can be realized by adopting an integrated approach usingcompression and encryption techniques, such a system is termed as crypto-compression system. Encryption is a coding technique that provides security whereas data compression is also a coding technique, whose purpose is to reduce both the data storage size and ultimately the time for data transmission. In this paper, an algorithm has been proposed which uses the compression and data encryption techniques. Firstly, data size is reduced through various compression techniques in order to increase the data transfer rate. Then the compressed data is encrypted to raise its security. Thus, technique proposed in this paper is useful in reducing data size, raising data transfer rate and providing security during communication. In this proposed system, encoded string is created from an input string of symbols and characters based on entropy encoding technique like arithmetic coding that can be used to achieve high level of compression in the present network topologies for exchange of data with more security and compression.

Keywords
Arithmetic coding, one-time pads, stream cipher cryptography, entropy encoding, crypto-compression system

1. INTRODUCTION
Information security (protection of information stored on standard disks, tape) is a growing issue among IT organizations of all sizes. To tackle this growing concern, more and more IT firms are moving towards cryptography to protect their valuable information. In addition to above concerns over securing stored data, IT organizations are also facing challenges with ever-increasing costs of storage required to make sure that there is enough storage capacity to meet the organization's current and future demands. To keep up with the challenges in association with the limited IT infrastructure budget and rising storage capacity needs, some enterpriseshave started using data compression techniques. An integrated solution to both-capacity and the security problems has been use of software-based compression techniques to reduce the number of disk increasing storage density and encryption techniques to secure the confidential data on the tapes.

1.1 Cryptography
Cryptography is the process of designing techniques to protect data. It provides security and integrity of confidential messages using different forms of encryption and decryption. Cryptography is a process of storing and transmitting data in a scrambled form so that only those for whom it is intended can read and process it. The term is often associated with scrambling plaintext (ordinary text) into cipher text (this process is called as encryption), then back again (known as decryption). Nowadays it's being used all around us- from ATM cards to ecommerce websites, in gaming consoles, for distribution of copyrighted music and film and many more applications.
Within the context of any application, there are some specific security principles requirements such as:-
• Authentication: -Securing one's identity.
•Confidentiality: - Ensuring that no other person can read the message except the intendedreceiver.
• Integrity: - Assuring that the received message has not been tempered in any way from the original.
• Non-repudiation:- A mechanism to prove to ensure that sender cannot disown this message.
There are two types of cryptography based on the number of keys used in encryption and decryption process-symmetric cryptography and asymmetric cryptography.

1.1.1 Asymmetric-key Cryptography
In asymmetric-key cryptosystems, the public key is distributed among senders of message by recipient, while its paired private key remainssecret to recipient. The public key is used for encryption, while the private (secret key) is used for decryption. This algorithm is also known as public key cryptography e.g. RSA, DSA, and PGP etc.
While Diffie and Hellmanm[1] showed that public-key cryptography was possible by postulating the Diffie–Hellman key exchange protocol (a solution that is widely used in secure data exchange to allow the two parties to secretly agree on a shared encryption key).Diffie and Hellman's publication sparked efforts in probing a practical implementation of public-key encryption system. This was finally published in 1978 by Ronald Rivest, Adi Shamir, and Len Adleman, (whose solution become known as the RSA algorithm.)

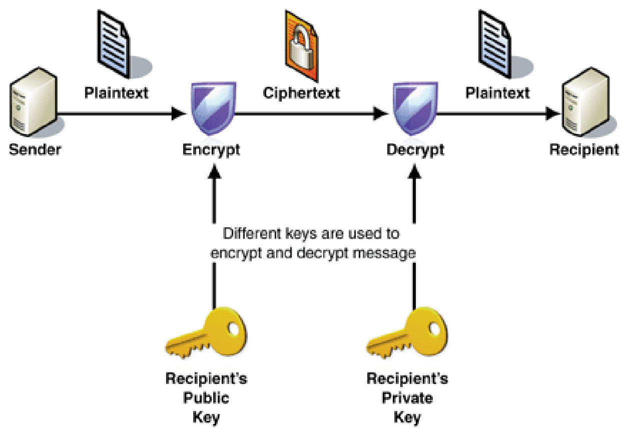

Figure 1: Asymmetric–Key Cryptography

1.1.2 Symmetric-key cryptography

Symmetric-key cryptography [1] refers to encryption technique in which both sender as well as receiver share the same keysecretly This was the only kind of encryption known until June 1976. These are implemented in the form of either block ciphers or stream ciphers. A block cipher enciphers blocks of plaintext rather than individual characters.The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are example of block cipher (designated cryptography standards by the US government).

Stream ciphers, in comparison to the block type, create a long stream of key of same length as the message, which is enciphered with the plaintext bit-by-bit e.g. one-time pad. In a stream cipher technique, the output cipher text is created using a hidden internal stage that changes as the cipher operates. This internal stage is initially set up using the privatekey. RC4 is a widely used stream cipher.

In this paper, one time pads are used as symmetric cryptography technique for encryption due to its simplicity.

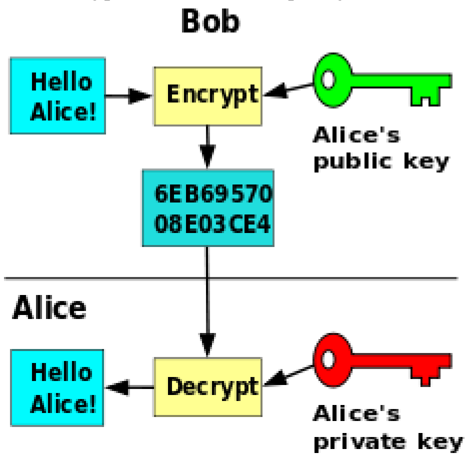

Figure 2: Example of Symmetric-Key Cryptography

1.2 Data Compression

While the history of data compression does not start as early as classic cryptography, the first model of data compression was developed in 1838.This early form was known as Morse code. The model improved in the mid-twentieth century when Information Theory [2] was introduced by Claude Shannon in 1949. It involved substituting smaller code-words for message content based on probability. In 1952, Huffman Coding optimized the practice of assigning smaller code-words to blocks with high probabilities of occurrence in a text. This idea was changed in the 1970s when it was realized that the assignment of smaller code-words need not be based on probability of blocks occurring, but rather code-words should be assigned dynamically, with text characteristics unique to a given piece of data.

In the beginning of the modern technology, storage capacity was very limited. Moreover, large file sizes were difficult to handle and cumbersome. Today, while storage capacity is exponentially increasing and large capacity storage device have become affordable but over the last decade there has been an unprecedented explosion in the amount of digital data transmitted via the Internetand thus the number of digital files are growing rapidly as well. For this reason, compression methods for digital files have come into more practical existence since the early days of computing. Methods of compression have always existed with computers but now are consistently improvising to allow for more data to fit on a single disk.In modern age, companies report that their consumer data is worth more than their physical company assets and there is a definite need to focus on how company's store data. In fact, more data requires more available space on the computer hard disk.

- Requires less disk space than an uncompressed file.
- Transfers faster than an uncompressed file.

The advantages of smaller files will always be evident:

Today, computer user interacts with compressed files every day. The field of data compression is massive and ubiquitous. However, much of the implemented compression today requires specialized hardware.The algorithms focused by us are implemented with common computer hardware on the average configuration machine.

Data compression also offers an interesting approach for reducing communication costs using available bandwidth effectively.Compression algorithms eliminates redundancy in data representation to decrease the storage required for that data [3].

There are two forms of data compression: Lossless and Lossy. The difference in them is related to the ability to get back the exact contents of the original file. One may think that compression always need to be lossless and should return the exact original contents, but in the case of images, videos, and audio files, the omission of many bytes is acceptable as it goes unnoticed by the human eyes. Text files like typed documents, source code and many more cannot afford to lose any bytes in data compression as critical functionality of the document may be lost.[4] This factor is controlled using:

$$\text{Compression Ratio} = \left(1 - \left(\frac{compressed\ size}{uncompressed\ size}\right)\right) * 100$$

Some of the ways of data compression are Huffman Coding, Arithmetic Coding etc. In this paper Arithmetic Codingis due to its added advantages over Huffman Coding discussed later in this paper.

1.3 Compression-Crypto System

While actual methods of cryptography vary but the process can be abstracted as:

$\text{Plain Text} \xRightarrow[Encryption]{} \text{Encrypted Data} \xRightarrow[Decryption]{} \text{Plain Text}$

This simpler form of cryptography parallels the process of Data Compression:

$\text{Plain Text} \xRightarrow[Compression]{} \text{Compressed Data} \xRightarrow[Decompression]{} \text{Plain text}$

Looking the summarized form of above two processes,it is clear that they serve different purposes. Yet compression results in a file incapable of being read until it undergoes a decompression process.This looks like a subtle form of cryptography. The similarities in these processes pose the question: Is there any commonality or an overlap? In other words: Is there a method of data compression that offers cryptographic features, or a form of cryptography that results in an encrypted data of smaller size than the original plain text?

If a compression method that requires a key to be decompressed be difficult to decompress without the key, then one may conclude that data compression is an appropriate form of cryptography. Conversely, if there is no form of data compression that offers cryptographic features, then there might not exist any overlap in these two processes that share similar methods[12].

The main similarity between data compression and cryptography exists within the compressed or encrypted file: For a human observer, this file is unreadable data, and thus useless until it undergoes a reversing process: decompression or decryption. However, it quickly becomes apparent that one is comparing apples and oranges. The priorities of each process are very different, as seen below:

Table 1: Cryptography vs. Data Compression Prioritized List

Cryptography	Data Compression
1.)Security (Integrity and Confidentiality) 2.) Practicality (Availability) 3.) Convenience (Speed)	1.)Compression factor 2.)Convenience (Speed) 3.)Security (Confidentiality)

Here, one can see that a combination method would require some form of compromise in either security or compression factor- ultimately this is detrimental to the end result. That is, if data is important then sender need complex encryption and file size is unimportant. If security is sacrificed for better compression ratio, then the security of the data does not never a main concern. Therefore, one can conclude that the two processes, despite their similar structure, do not aim to complement each other, so their direct combination can be cumbersome. In fact, tremendous complexity in modern cryptographic methods leaves no space for an attempt to compress data- additionally, if a compression method is somehow injected into the encryption algorithm, there would be a known structure and similar pattern in the cypher text–which might deduce the original encryption. However, in a review of the Pretty-Good-Protection routine, researchers were less successful in cracking encryptions that were first compressed.

This vagueness in answer to above questions suggests that there is still development opportunities to connect these fields.

There also occurs a problem of the sequence of applying these two processes i.e. Compression should be applied first before Encryption or Encryption should be applied first before compression.

Encryption can also be applied before compression in some particular situations such as when it is desired to transmit redundant data over an insecure and bandwidth-constrained channel [5]. Traditionally, data from a source is first compressed and then encrypted before it is transmitted over a channel to the receiver. In many cases this approach is benefitting, but there exist scenarios where there is a need to reverse the order of data encryption and compression. Consider a network of low cost sensor nodes that transmit confidential information over the internet to a recipient. So sensor nodes need to encrypt data to hide it from potential intruder, but they do not necessarily want to compress it as that would require additional hardware cost. On the other hand, the network operator responsible for transferring the data to the recipient wants to compress the data in order to maximize the utilization of its resources. It is important to note that the network operator might not be trustworthy and hence is not provided access to the key used for encryption and decryption of data.

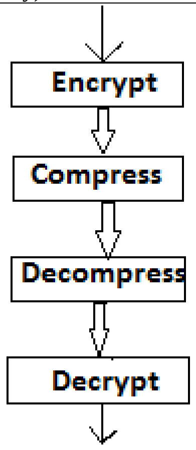

Figure 3:Flow-Chart for Encryption before Decryption Process

Compression should be applied first before encryption [5]because of the following reasons:

- Compressing data last won't reduce the file size much- Good encryption makes any input data (like redundant data) appear random. But compression works by removing redundancy and won't work well on random data.
- Compressing data first should decrease the effectiveness of many attacks-Compression works by removing the redundancy in the data. A common cryptanalysis (breaking cipher) method relies on finding repeated data. Compressing data should reduce its effectiveness.
- Brute force attacks will take longer- Brute force attacks work by trying out various possible keys combination and checking if the output data makes any sense. By compressing data first, an attacker has to decrypt the data and then decompress it before seeing if the output data makes any sense. This takes much longer, and if an attacker doesn't know sender's compressing the data at all, he/she might never be able to deduce the input.
- The opponents get less exposure to cipher text for analysis- Lesser the data the enemy has to analyse, the fewer hints they have about the internal state of sender's cipher and its key.
- Compressing a file after encrypting it is inefficient- Cipher text produced by a complex encryption algorithm generally has a uniform distribution of characters. As a result, a compression algorithm will be unable to find high percentage of redundant patterns in such text and as a result there will be little data compression. In fact, if data compression

algorithm is able to significantly compress the cipher text, then this indicates a high level of inefficiency in the cipher text (evidence of poor encryption).

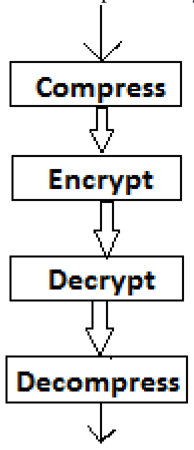

Figure 4: Flow-Chart for Compression before Decryption Process

2. LITERATURE & SURVEY

In the last decade, an unprecedented explosion of textual information occurred through the use of the Internet, digital library and information retrieval system.By the year 2004 the National Service Provider backbone had an estimated traffic around 30000Gbps and that the growth continued to be 100% every year. Currently, text data competes for 55% of the total internet traffic. Some of the existing system used compression along with RSA algorithm for mobile communication.Such systems were able to provide a solution to SMS security problem [6]. The solution that is used in this system is to compresses the SMS text in order to reduce its length, then encrypt it using RSA algorithm. But RSA is a Public Key Encryption method. A disadvantage of using public-key cryptography for encryption is speed. This system is known as Hybrid Compression Encryption (HCE) system. One more exiting system uses crypto-compression to provide us an errorless secure transmission of medical information data like Image, Audio, Video files of patient etc. This system uses lossless compression technique like Sequitur for efficient bandwidth utilization of communication channel. The combination of McEliece public-key cryptosystem with compression provides confidentiality in the transmission. But this system has a limitation with length- its efficiency drops as the data length increases. Also such system requires very large public key which makes it very difficult to adapt in many practical situations.

Now a day's Arithmetic coding is commonly used. It is a statistical method whose compression ratio is very good. Therefore, in this paper Arithmetic coding is used with private key encryption system because private key encryption system is fast with ease in mathematical implementation.

3. PROPOSED WORK

The proposed algorithm is based on the concept of entropy encoding like arithmetic coding in which each word of text file is converted into floating point number (lie in range between 0 and 1).Then this floating point number is converted into binary number and after that symmetric key is used to encrypt this binary number. Finally after encryption, resultant is again a binary number. This binary number is again converted into decimal number again and sends to the receiver.

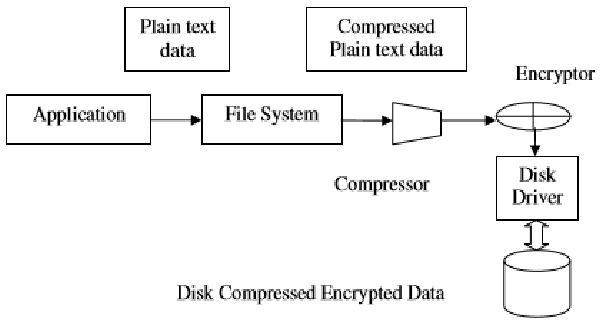

Figure 5: Crypto Compression Model

3.1Arithmetic Coding

It is known fact that the Huffman coding generates minimum redundancy codes compared to other algorithms. But the disadvantage of Huffman is that- it produces codes of the encoded data of different sizes. Therefore it becomes very difficult for the decoder to know that it has reached the last bit of a code. On the other hand, arithmetic coding (viewed as a generalization of Huffman coding) efficiently represents more frequently occurring sequences of pixels values with fewer bits. Moreover, arithmetic coding typically has a better compression ratio than Huffman coding because it produces a single symbol rather than several separate code-word.

To achieve better compression factor than Huffman coding, arithmetic coding uses decimal numbers to represent text. Arithmetic Coding is a lossless compressioninvented by JormaRissane and turned into a practical method by Witten, Neal and Cleary [7]. It is not restricted by the bounds of integer expression in computers(requiring rounding all decimal numbers to the nearest whole integer to be represented as bits) rather arithmetic coding accepts real numbers [8]. Arithmetic coding defines a range for each character in a word based on its frequency occurrences. Arithmetic coding encodes the entire word into a single number (fraction n where $0.0 \le n < 1.0$)[9]. The coding algorithm is symbol wise recursive i.e. it executes upon and encodes (or decodes) one symbol per recursive cycle.On each iteration, the algorithm partitions an interval between 0 and 1 and retains one of the partition as the new interval. The word is recovered by magnitude comparison of the code to recreate the scenario when the encoder must have successively partitioned and retained each subinterval.

Let us see an example, suppose the word-HELLO is to be encoded. Arithmetic Coding is fast on a computer but can be very slow manually, so word HELLO is chosen because it has 5 total letters (4 unique letters), so it can show compression. Firstly,computation of the probabilities and intervals of each character in the given word is to be done, the lowest possible interval for a character to occupy in HELLO is 1/ 5 (because there are 5 characters). A character's interval length in this word equivalent to 1/5*Total Occurrence's.[4]

Table 2: Arithmetic Coding Intervals for HELLO

Character	Probability	Interval
H	0.20	0.00-0.20
E	0.20	0.20-0.40
L	0.40	0.40-0.80
O	0.20	0.80-1.00

To encode any string, the total interval (0−1) is divided repeatedly and the character's high and low interval values are added to the encoded value in the order that the encoder reads each letter. This process is difficult to follow manually.

HiVal⟵ 1.0 /*Upper limit of Interval*/

```
LoVal⟵ 0.0                    /*Lower limit of Interval*/
WHILE (more characters to process)
        Char ⟵ Next message character
        Interval ⟵ HiVal – LoVal
        CharHiVal⟵ Upper interval limit for char
        CharLoVal⟵ lower interval limit for char
        HiVal⟵ LoVal + Interval * CharHiVal
        LoVal⟵  LoVal + Interval *CharLoVal
END WHILE
OUTPUT (LoVal)
```

Pseudo Code for Arithmetic Coding

Following this pseudo-code, the proceeding table is generated with the referenced charLoVal, charHival, LoVal, and HiVal variables:

Table 3: Arithmetic Coding of HELLO

Character	Interval	Char LoVal	Char HiVal	LoVal	HiVal
				0.00000	1.00000
H	1.0000	0.00	0.20	0.00000	0.20000
E	0.2000	0.20	0.40	0.04000	0.08000
L	0.0400	0.40	0.60	0.00160	0.00240
L	0.0080	0.60	0.80	0.00480	0.00640
O	0.0016	0.80	1.00	0.00128	0.00160

Encoded Message: 0.00128

The final LoVal is the encoded message, in case of HELLO encodes to 0.00128. While difficult to follow arithmetic encoding algorithm on paper, it is easily implemented in a computer. Decryption is similar, reverse looking to match successively divided ranges with character high and low values in frequency. The important feature of arithmetic coding is that it does not represent real numbers with integer values, but rather can represent them exactly with decimal values. However, error checking is critical in arithmetic encoding compression/decompression algorithms because computers see arithmetic encoded data as numbers in floating point representation. This leads to potential problems of underflow in addition and sometimes missing a zero condition. However if errors are kept in, arithmetic encoding obtains better compression ratios as it can represent exact probabilities. It is far slower than Huffman Encoding due to extra efforts required when dealing with floating point numbers in the computer.

3.2One Time Pads

The method was presented in 1918 by Gilbert Vernam and Joseph Mauborgne. One time pads have a unique cryptographic quality that it cannot be broken because cypher text contains no information about the key other than its length. The basic principle is to create a random key of the same length as the plain text message and add the two together, character by character:

It is important to note that each plain text L translated to a different cypher text character: D,P,Z etc. randomly: there is an equal probability that these characters could translate to any other character or not even change at all.

Decrypting the message: simply subtract the key from the cypher Text.

[10]When implemented in a computer, the message and key are simply bit strings that undergo an EXCLUSIVE OR operation for encryption. This is then just done again between the cypher text and key to decrypt.

For a one time pad to be truly unbreakable, the key must be generated randomly and never used again (‘one time’). This creates difficulties in the practice use of a onetime pad because a random key would require rolling a 26-sided dice for every character in the key.If the key is not truly random or is based on a pseudo random number generator, then there exists the possibility to regenerate the same key. Once the random key is created, it must be sent in full to the recipient along with the message because without the key, the cypher text is unbreakable,

In summary, one time pads have the ability to be completely un-crack able, but require a random key of equal length to the original message.

Message	H	E	L	L	O	W	O	R	L	D
Key +S	G	S	E	G	F	A	I	O	T mod 26	
Cypher Values	25	10	3	15	10	1	14	25	25	22
Interpreted as letters	Z	K	D	P	K	B	O	Z	Z	W

4. DESIGN

4.1Compress and Encryption of message

Step 1:-

- Initialize Lower limit=0.0, Upper limit=1.0
- While there are still symbols to encode
- Current range = Upper limit - Lower limit
- Upper limit = Lower limit + (Current range * Upper limit of current symbol)
- Lower limit = Lower bound + (Current range * Upper limit of current symbol)
- End while

Step 2:-

- The string may be encoded using any value within the range and after that the output decimal number is converted into binary format.

Step 3:-

- Limit the number of bits obtained in step 2 by using the formula:-
  No of bits=log (┌2/upper limit last encoded symbol - lower limit last encoded symbol┐)

Step 4:-

- Select randomly any one time pad of message length (Compressed message) andXOR it with result of step3.

Step 5:-

- Rotate 2 (or more) bits right depending on requirement.

Step 6:-

- Convert the result of step 5 into decimal format again.

Output: - Output is floating point number corresponding to the input symbols.

4.2 Decompression and Decryption of message

(Convert the floating point number received into original text Algorithm)

Step 1:-

- Convert the received floating point number into binary format.

Step 2:-

- Rotate 2 bits (or more depending on encryption algorithm) to left.

Step 3:-

- Selected one time pad is XORed with the result of step2

Step 4:-

- Convert the result back into decimal form.

Step 5:-

- Encoded value=Coded input
- Until string is not fully decoded do-
- Identify the symbol containing the encoded value within its probability range
- Current range = upper limit of current symbol - Lower limit of current symbol
- Encoded value = (encoded value - Lower limit of current symbol) /current range
- End while

Output: The output is the original symbol

4.3 Example

Table 4: Arithmetic Coding Intervals

Symbol	Probability	Range (lower bound, upper bound)
a	30%	(0.00,0.30)
b	15%	(0.30,0.45)
c	25%	(0.45,0.70)
d	10%	(0.70,0.80)
e	20%	(0.80,1.00)

Compression and encryption-
Data to be encoded and encrypted is "abd"
Step1:-
Encode 'a'

- Current range= 1 - 0 = 1
- Upper bound= 0 + (1 × 0.3) = 0.3
- Lower bound= 0 + (1 × 0.0) = 0.0

Encode 'b'

- Current range= 0.3 - 0.0 = 0.3
- Upper bound= 0.0 + (0.3 × 0.45) = 0.135
- Lower bound= 0.0 + (0.3 × 0.3) = 0.09

Encode'd'

- Current range= 0.135-0.09 = 0.045
- Upper bound= 0.09 + (0.8 × 0.045) = 0.126
- Lower bound= 0.09 + (0.7×0.045) = 0.1215

Step 2:-

- The word-abd may be encoded using any value within the range (0.126, 0.1215).Now output is 0.12375(arithmetic mean of lower and upper bound) and its binary equivalent= .00011111101011100001010001.

Step 3:-

- No of bits= log ┌2/0.0045┐ = ┌log444.44┐ =8bits

Step 4:-

- So after reducing number of bits binary value is 0.00011111.

Step 5:-

- One time pad is – 11010100
- Data- 00011111 from step 4.
- After XORing the output is- 11001011

Step 6:-

- Rotate 2 bits right the result is 11110010

Step 7:

- 0.11110010 in decimal is 0.96875

Decompression and Decryption-
Step 1:-

- Received data is 0.96875and binary format of received data is 0.11110010

Step 2:-

- Apply 2 left shifts to result of step1 the result is 11001011

Step 3:-

- Apply selected one time pad and Xored it with the result of step2 the result is 00011111

Step 4:-

- Convert .00011111 into decimal i.e. 0.1210

Step 5:-

- Decoded first symbol 0.1210 is within (0.00, 0.30) 0.1210 encodes 'a'
- Remove effects of 'a' from encode value i.e. Current range = 0.30 - 0.00 = 0.30
- Encoded value = (0.1210 - 0.0)/0.30 = 0.40330
- Decoded second symbol 0.40330 is within [0.300, 0.450) 0.4033 encodes 'b'
- Remove effects of 'b' from encode value i.e. Current range = (0.45 - 0.30) = 0.15
- Encoded value = (0.4033 - 0.30)/0.15 = 0.6886
- Decoded third symbol 0.6886 is within (0.70, 0.80) 0.6886 encodes'd'

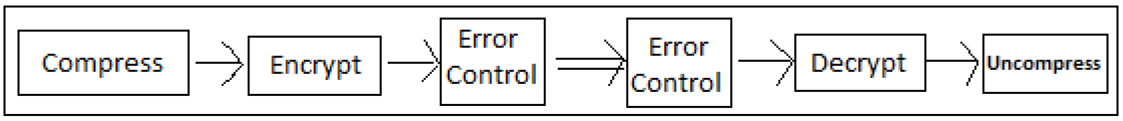

Figure 6: Crypto Compression System Flow Chart[11]

5. EVALUATION

The proposed technique has the following key features:

- It provides precision control to convert entire string or file.
- It uses both data Compression and Cryptography technique. Therefore conversion timeof compressed as well as encrypted data (generated through crypto-compression system) to plain text and vice-versa is usually high as compared to encrypted text conversion to plain text or compressed text conversion to plain text.
- It is a highly secured way of transmission ascipher text generated for same information always different due to one- time pad usage during encryption. Moreover,it is a private key encryption technique.

For example-suppose we are given three file (.txt) with following sizes:

Table 5: Conversion Time for Various Size Text Files

File (.txt)	Size (in KB)	One Time Pad (in sec)	Huffman Coding (in sec)	Proposed Technique (in sec)
File 1	135	0.31	0.75	1.02
File 2	670	2.00	3.50	4.50
File 3	1380	5.60	7.10	9.20

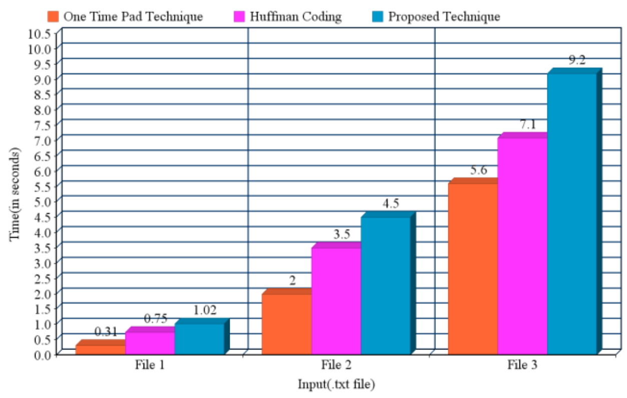

Figure 7: Conversion Time Comparison

- In an ideal channel, the reduction of transmission time is directly proportional to the amount of compression. Therefore transmission time of data compressed using Huffman coding only and data generated through the proposed system are almost equivalent. But when compared to data transmission after one time padding (encryption)this does not hold true. Clearly Huffman coding generated data and the proposed system generated data are clear winners in this aspect.

Table 5: Transfer Time for Various Size Text Files

File (.txt)	Size (in KB)	One Time Pad (in sec)	Huffman Coding (in sec)	Proposed Technique (in sec)
File 1	135	2.30	1.80	1.70
File 2	670	7.90	5.10	4.92
File 3	1380	13.70	9.50	9.15

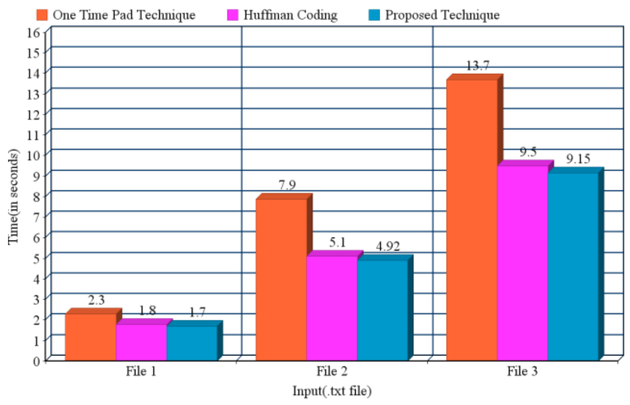

Figure 8: Transfer Time Comparison

- In short, we can say that proposed technique generated cipher text takes less bandwidth of channel when compared to one time pad whereas when it is compared to Huffman coding, bandwidth requirement of both the technique are almost same but proposed technique is much more secured way of transmission.

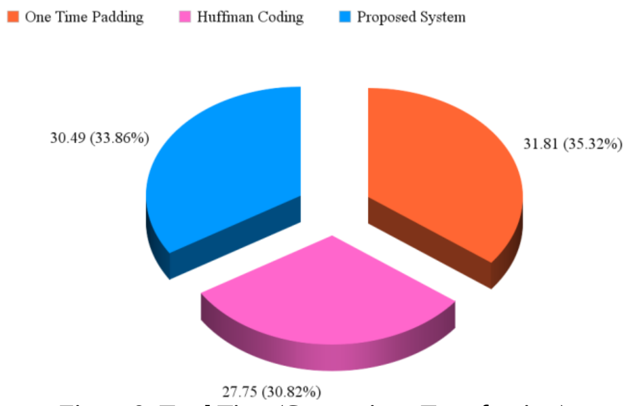

Figure 9: Total Time (Conversion +Transfer time) Comparison

6. CONCLUSION

This analysis answered the original question- Is there any efficient method of combining encryption and compression? Answer is simple: It will be self-defeating to compromise. The two processes may have different priorities and combining those compromises and desegregation the quality. However, evaluationdone through this paper suggest that when the processes are done in series: files are first compressed and then encrypted, the processes can work off each other, for a better result.

The proposed technique provides an excellent integration of data compression with the cryptography to increases the data security and transfer rate during data transmission. Size of data is reduced using the arithmetic encoding data compression technique and after that compressed data can be encrypted to provide the security. The Present network scenario demands exchange of information with reduction in data storage and time for data transmission along with security. Technique proposed in this paper fulfils above requirements as this technique use the concept of data compression and encryption.

7. ACKNOWLEDGMENT

We would like to express our gratitude and a special thanks to the editors and the reviewers for giving very insightful and encouraging comments.

8. REFERENCES

[1] http://en.wikipedia.org/wiki/Cryptography

[2] Shannon C. E. “A Mathematical Theory of Communication”. The Bell System Technical Journal, Vol. 27

[3] Dr. V.K. Govindan and B.S. Shajeemohan -“An intelligent text data encryption and compression for high speed and secure data transmission over internet”

[4] Null, Linda, and Julia Lobur. Essentials of Computer Organization and Architecture. Sudbury, MA: Jones & Bartlett Learning, 2012

[5] Dr.Mukesh Sharma and Smiley Gandhi- “Compression and Encryption: An Integrated Approach” International Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 5, July – 2012

[6] Tarek M Mahmoud, Bahgat A. Abdel-latef, Awny A. Ahmed and Ahmed M Mahfouz -“Hybrid Compression Encryption Technique for Securing SMS”, International Journal of Computer Science and Security (IJCSS), Volume (3): Issue(6)

[7] I.H. Willen, RandfordM.Neala and John G.Cleary - “Arithmetic Coding for Data Compression”, Communications of the ACM Volume 30 Issue 6.

[8] V.Kavitha and K.S Easwarakumar,“Enhancing Privacy in Arithmetic Coding” ICGSTAIML journal, Volume 8, Issue I,2008

[9] J.A Storer-“Data Compression: Methods and Theory” Computer Science Press.

[10] Trappe, Wade, and Lawrence C. Washington. Introduction to Cryptography: with Coding Theory. Upper Saddle River, NJ: Prentice Hall, 2002.Print

[11] Schneier, Bruce. Applied Cryptography: Protocols, Algorithms, and Source Code in C. New York: Wiley, 1996. Print

[12] H. Kruse and A. Mukherjee. - “Data Compression Using Text Encryption”, Proc. Data Compression Conference, IEEE Computer Society Press, 1997