Review on Optimizing Compression Performance in Search Based Chaotic Cryptosystem

Veenus P K
Research Scholar, Dept. of CSE,
Noorul Islam University, Kumaracoil,
Tamil Nadu, India

Vivek P K
Electrical Section, Dept. of Engg
Ibra College of Technology
Ibra, Sultanate of Oman

K Sivasankar
Assistant Professor, Dept. of IT,
Noorul Islam University, Kumaracoil,
Tamil Nadu, India

Remya T P
Teaching Faculty, Dept. of General Education
Govt. of Kerala,
Kerala, India.

***Abstract*— The conventional encryption methods are not appropriate for image cryptography as they cover huge volume of data and the pixels have a strong correlation. Currently, data encryption and encoding have become more critical and challenging, in multimedia communication. The rapid developments in digital communication have made wide use of multimedia data. The basic characteristics of multimedia communication system like security, high transmission rate, low bandwidth, redundancy of data and low storage capacity makes basic encryption and compression algorithms mandatory. This paper discusses the basic Baptista chaotic system and its variants made by incorporating Huffman encoding. This image encryption algorithm optimizes the compression performance in Baptista's chaotic system. Result analysis proves that the compression performance and encryption speed of the modified system is much better. This paper made a proposal for implementing this optimization algorithm in cloud computing. So that any organization can exploit the cloud computing as an optimization framework for the large amount of data sharing by preserving better security and compression performance.**

Keywords*— *Encryption, Cloud computing, Compression, Cryptography, Chaos, Logistic map.

I. INTRODUCTION

Cloud computing is a group of services including the hardware and operating system infrastructure and virtualization components. Cloud architectures are established in an on-demand manner. That is, the resources are dynamically assigned to a user according to his request, and surrendered after the job is done. Cloud computing enables us to access shared resources and manipulation of large amounts of stored data over public or private networks. By the optimization features like resource and data sharing, it became very popular in the internet community. It helps to reduce the cost of IT infrastructure and maintenance by meeting dynamic unpredictable demand on resources. In cloud computing, information is distributed over the network. So the security of these large amounts of data become a main challenge in this field. To secure the data, it has to apply some cryptographic algorithms.

In the present era of multimedia communication networks, cryptography has achieved special consideration. It is the science of providing the security features of information in any communication system. Traditional cryptographic techniques are based on the number theory. Pecorra and Carrol discovered chaotic synchronization principles. The chaotic system has the manners of a nonlinear dynamic system, which seemingly appearances as random [4]. But it is resulting from the defining deterministic processes for particular system parameters. The basic properties of chaotic schemes [11] are sensitivity on initial conditions and control parameters, which provide secure communication methods. Nowadays chaos based cryptographic systems seems like a substitute to the existing techniques such as the Rivest-Shamir-Adleman algorithm (RSA), Elliptic Curve Cryptography etc. The chaotic cryptographic scheme can be implemented in the simplest form of one or two dimensional systems represented by discrete chaotic maps. This chaotic method can generate complicated dynamics with the use of chaotic digital maps. This is desirable in cryptographic algorithms.

There exists a strong linking between cryptography and chaos. Unpredictable chaotic orbits are generated with the deterministic equations of chaotic systems, as they are extremely sensitive to its initial conditions. The chaotic properties like sensitivity to initial values, ergodicity, mixing, etc., are comparable with the confusion-diffusion properties of conventional cryptography.

By the sensitivity nature of chaos to initial conditions, initially very near trajectories may diverge exponentially in a small period of time. Without the initial information about the system, long term forecast of the upcoming statuses of the chaotic system is impossible. By the ergodicity property, a path in phase space comes arbitrarily near to its former status. This reflects that the system is eventually limited to its spatial object made up of an attractor. The density of this attractor is time invariant. By mixing property, a small interval of the preliminary state of a chaotic system spreads over the full phase space in its growth.

The basic cryptographic algorithms are concerned only about the security features of data over the communication network. Multimedia data like image, consists of large volumes of data and so there arise the importance of

compression. Implanting compression in cryptographic algorithms [2] attained much consideration in this area. While maintaining the security, this will provide compression performance also. Baptista technique [9] possess a major disadvantage [12] of low compression performance. So the Huffman encoding algorithm is used in embedding compression in Baptista method.

In recent years, compressing the files before transmitting has gained a lot of interest with a fast growth of multimedia data through an open network. The method of decreasing the size of a data file is often denoted as data compression. Compression techniques are used to decrease the quantity of data for saving the storage space volume. It is called source coding as the data source is encoded before transmission. Compression is advantageous because it reduces resources essential to store and transmit data.

In data compression, information is represented with less bits than the original. Compression is of two types, lossless or lossy. Lossless means no data is missing in the decompression, while it is kept in the compressed format. Lossy means some data is lost at the time of decompression. Lossless data compression is classified into Run-Length, Huffman and Transformation. Transformation is again divided into Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT). Lossy data compression is classified into Joint Photographic Experts Group (JPEG), Moving Picture Experts Group (MPEG) and MPEG Audio Layer 3. Huffman coding is a variable distance coding. Since it provides increased compression rate, it is commonly used as a compression technique during transmission of data.

The rest of this paper is organized into 4 sections. The section 2 will explain the existing methods of cloud computing, encryption and compression method. The section 3 deals with the integration of compression methods in the search based chaotic encryption scheme. This explains in detail about the new variant of Baptista method proposed by Wong et al. The section 5 deals with the result analysis.

II. EXISTING METHOD

A. *Cloud Computing*

Cloud computing is a technology which enables universal access to shared data and services [1] through the internet. Cloud computing became very popular with the large network facility and other service oriented environment. Any organization can take benefit from this optimizing technology. By optimizing IT infrastructure, it helps the company to cut costs of the organization. Virtualization technology enables cloud computing to increase the speed of computing through the large infrastructure utilization. Cloud computing is a type of grid computing. Other than the conventional parallel computing, it provides a dynamic platform for the parallel applications with reasonable price.

However, security aspects of the data in the cloud computing environment need much more concern [3]. Since large amounts of data is shared through an open network. Lack of protection of the data is the major challenge in the cloud computing [13]. The complexity of dealing with the information security makes uncertainty in this field. The best solution for this problem is the introduction of encryption algorithms in this area. Therefore, this paper proposed to implement the encryption algorithm in a cloud computing system. The encryption can prevent unauthorized data accessing in the open network. This is one of the best and most popular solutions for securing confidential information. Other than the conventional encryption method, the search based chaotic encryption algorithm can provide better security features.

B. *Encryption*

In 1998, M.S. Baptista had proposed a chaos-based crypto system which became very popular in the area of chaotic encryption. This method, inputs a text data composed by some alphabet as the plaintext. Taking the benefit of ergodicity property, the ciphertext is created from the plaintext. The system is limited to a group of points called an attractor. The whole of the attractor is divided into S sites. In this method, S = 256. The size of each site can be calculated using equation (1).

$$\varepsilon = (x_{max} - x_{min})/S \quad (1)$$

Where x_{max} and x_{min} are the corresponding upper and lower value of the attractor. The entire phase space of the given map is distributed into a number of equal size partitions. Each partition, ε interval, map to a plaintext symbol. The partitioned phase space along with the corresponding plaintext mapping is considered as a lookup table for encryption. A pseudo random sequence is generated by a chaotic map is used for searching in the lookup table. Baptista method uses logistic map as the chaotic map. This is a one-dimensional map proposed by R.M. May as in equation (2).

$$x_{n+1} = bx_n(1 - x_n) \quad (2)$$

Where $x_n \in [0,1]$ is the output at discrete time n=0, 1, 2, etc and control parameter b should be a real number between 3.6 and 4. The above equation provides its trajectory, leaving from an initial condition x_0 reach an ε-interval associated with that character. Fig. 1 shows a schematic representation of the way of associating the S units alphabet with the $S\varepsilon$ intervals.

A secret chaotic trajectory is generated from the parameters and initial condition. This will search the corresponding part of the plaintext symbol which has to be encrypted. The number of iterations of the logistic map will be the length of the searching trajectory. This number of iterations will be the ciphertext.

In the decryption process, the similarly secret chaotic search path is redeveloped. It can recover the plaintext symbols with the identical secret key and the lookup table. The Baptista-type search based chaotic crypto system has low compression performance. The ciphertext is generally around 1.5 to 2 times of the plaintext length. So it needs to implement a new method which has better compression performance than the Baptista method. In the following section, a Huffman

encoding scheme is being discussed. Reference [5] used this for incorporating compression in Baptista method.

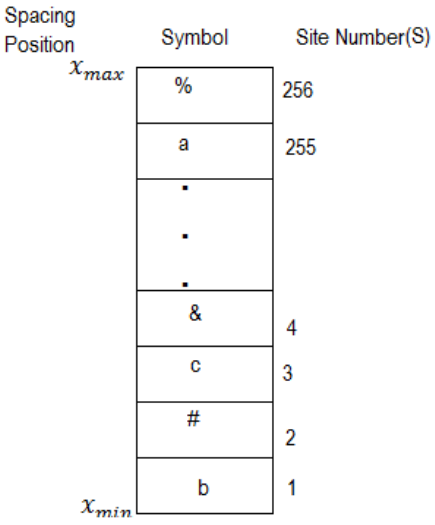

Fig. 1. Division of S sites in Baptista method

C. Compression

The In 1952, David Huffman developed a lossless Huffman coding system for data compression. Based on the possibility of occurrence of every symbol, it will generate a variable length code. It reduces the size of data by giving a short code for more possible symbols and longest code for less probable symbols. In the 1940s, Claude Shannon and Robert Fano proposed independently a coding method known as Shannon-Fano method. They constructed codes from the topmost to bottom, but in Huffman coding it is from the bottom to top.

Huffman coding technique consists of simple steps of sorting and merging of symbols. It constructs a binary tree starts from the bottom. At first it wants to sort all symbols based on the chances of occurrence in descending order. In the beginning all the symbols are considered as the leaf nodes. A binary tree is built through several steps. In each step, two smallest possible symbols take into consideration and combine them into a single one. Add probabilities of each symbol and it will be the possibility of the combined one. Continuing like this, at the end there will be only one symbol. All the symbols are merged to become one single symbol. To calculate the code words of each symbol, it wants to traverse the binary tree from root to a leaf node. It requires to assign arbitrarily a 0 to the left edge and 1 to the right edge.

TABLE I. PROBABILITY OF SYMBOLS

SI No	Symbols and Probability	
	Symbols	*Probability*
1	A	0.40
2	B	0.35
3	C	0.125
4	D	0.125

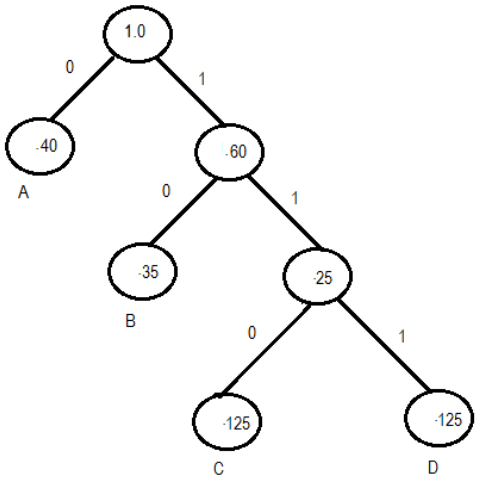

Fig. 2. Huffman coding

The process can be explained with an example as in Fig. 2. As in the Table I, four symbols are given with their equivalent probabilities. At first, sort the symbols with their probability of occurrences in descending order. Here C and D have smallest probability of 0.125. So it needs to take that two symbols as leaf nodes and add its probability. So 0.25 will be the frequency of the new merged symbol CD. The symbol B has the probability 0.35. So merge B with CD to get new merged symbol BCD with probability 0.60. This BCD can combine with the symbol A to get ABCD as the final root node with a probability 1.0. Thus the binary tree will be completed at the bottom to up manner. By assigning 0 bits at the left edge and 1 bit in the right edge, the code words of the four symbols can be determined as in the Table II. Average size of the code word, i.e. Bit rate (R) can be calculated by (3).

$$R = \sum_{i=1}^{n} P_i l_i \quad (3)$$

Where P_i is the probability of each symbol and l_i is the length of the code word of each symbol.

R =. 4*1+.35*2+.125*3+.125*3=1.85 bits/symbol.

TABLE II. CODE WORDS OF SYMBOLS

SI No	Code words and Symbols		
	Symbols	*Code*	*Length*
1	A	0	1
2	B	10	2
3	C	110	3
4	D	111	3

The basic Huffman coding method is static and it will take too much time for compression. To generate the code word of all the symbols, it may take two passes. One is for counting the frequencies of entire symbols and other is for the compression. To solve this problem, the dynamic Huffman coding is developed by Faller and Gallager with substantial improvements by Knuth.

In this, the Huffman tree will be empty at the initial state. It will modify the tree in each step while reading the symbols. The first symbol in the input is added to the binary tree and will allot a code for it. At the time of the next visit, the current code is written to the output and the frequency will be incremented by one. This will modify the tree and check whether it is still a Huffman tree. Otherwise, it requires rearranging. To modify a tree, check each time, while inputting a symbol. Gallager introduced a theorem by which a full tree is said to be Huffman, if it satisfies the sibling property. The frequency of symbols in each level from left to right and from bottom to top, must be sorted in non-descending order. Thus the root has the highest frequency and bottom left has the lowest probability.

The basic algorithm consists of comparison and swapping of symbols. At first it compares the probability of the present node to the succeeding nodes. If the probability of the present node is lower than its successor, there won't be any change. Otherwise, it requires to swap with the one which has the smallest frequency. If there exists more than one node with small identical frequency than the present node, swap it with the last node in the group of the same frequency. The current node should not be swapped with its parent node. Then increment the occurrence of the current node by one. Correspondingly increment the frequencies of its parents. Until the current node becomes the root node, the process has to be repeated.

1) Adaptive Huffman Coding

The adaptive Huffman coding [10] can be explained in detail with an example as in Fig.3. The tree contains four symbols. A, B, C and D and its frequencies. 7 symbols in the tree are already processed. Here symbol A is taken as the present node and the modification of tree considers the updating status of only symbol A. At each step, checking of sibling property of binary tree has to be conducted. If it satisfies the property, it can say that it is a Huffman tree.

Left binary tree in Fig.3 shows the present frequency level of 4 symbols and right binary tree shows the final binary tree. In the first step, frequency of symbol A is incremented by 1 and it becomes 2. Now it satisfies the sibling property. So it does not want swapping. The only change in the tree is the corresponding increment in all the parent nodes of A. In the next step, frequency of A becomes 3. Frequency of A exceeds its successor B, C and D. Symbols B, C and D have the same frequency. So it needs to swap with the last one of the group of similar frequency. So this needs a swapping of D and A. Otherwise, it violates the sibling property.

In the next step, frequency of A is incremented from 3 to 4. Since all nodes satisfies the sibling property, swapping is not required. Now the frequency of A is 4 and which is equal to that of its successor in the immediate top level. So they should be swapped. In the succeeding step, frequency of A is incremented from 4 to 5. This will be the final tree, for the updating of frequency level of symbol A.

The adaptive Huffman tree can be implemented in the process of reading the file itself. It offers the facility to update the tree dynamically. So there is no time wastage in the process of compression. While in static model updating, it requires two passes – one for reading the entire input and one for creating the binary tree. So in practical applications, it is better to use the adaptive Huffman encoding rather than the static one. There are so many variations proposed by recent researchers for the adaptive Huffman encoding scheme. The idea behind all of them are same. Wong et al incorporated this compression technique in the Baptista method to improve the compression performance.

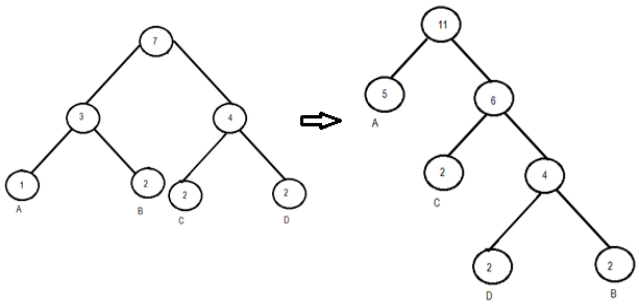

Fig. 3. Adaptive Huffman coding.

III. PROPOSED METHOD

Reference [7] proposed some variation of Baptista type search based chaotic crypto systems. Baptista used a static lookup table for the encryption, while a dynamic lookup table [6] based on the possibility of occurrence of the symbols are used in the modified scheme. As in the basic Baptista system, it maps the plaintext in a lookup table. The difference is that the former is static and the latter is dynamic. At first, scan the whole plaintext symbols and find out the frequency of every symbol. The whole phase space is divided and based on the chances of occurrence of the symbol, each one is assigned into the partitions. It can be better explained with an example as follows. Take 5 symbols A, B, C, D and E. Each one will be assigned the partition as per the frequency in the table. Total 256 partitions are there on a lookup table.

TABLE III. MAPPING OF SYMBOLS

SI No	Partitions based on probability of Symbols		
	Symbols	*Probability*	*No: of Partitions*
1	A	0.5	128
2	B	0.25	64
3	C	0.125	32
4	D	0.0625	16
5	E	0.0625	16
		Total	256

As given in Table III, based on the possibility of occurrence of the symbols, here 128 partitions will map to A, 64 will map to B, 32 will map to C and 16 each will map to both the symbol D and E. The corresponding dynamic lookup table is shown by Fig. 4. In the following step, it has to be checked for the plaintext symbol which is in the corresponding lookup table or not. If it is there, iterate the chaotic logistic map up to which it will reach the relevant parts of the dynamic lookup table. Here it also uses the logistic map as the underlying chaotic map to produce a

pseudorandom sequence as in Baptista method. $x_n \in [0, 1]$ is the output at discrete time n = 0, 1, 2 etc. and the parameter b should be a real number between 3.6 and 4. The control parameter, b and the initial condition, x_0 are taken as the secret key in this encryption method. Since it is a symmetric key encryption scheme, it uses the same key for both the process of encryption and decryption.

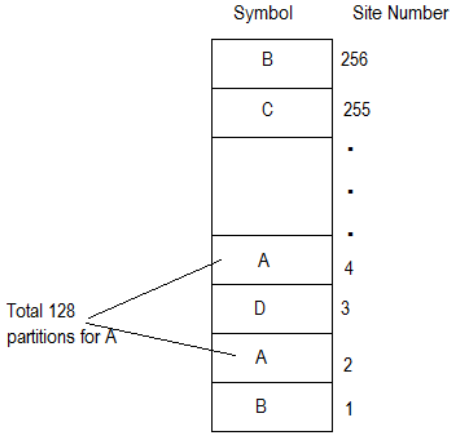

Fig.4. Dynamic Lookup Table

If the searching in the dynamic lookup table finds a relevant symbol, the number of repetitions by which it hit that symbol will be its corresponding ciphertext. If the plaintext is not in the dynamic lookup table, that kind of less probable plain text symbols will be encrypted only by chaotic mask mode. Otherwise, it will be scrambled by the mask mode. The plaintext character is taken as output directly, and the chaotic logistic map is iterated only once. To identify the use of mask mode for the present symbol, a special symbol is chosen as zero number of iterations, is employed since the number of repetitions in the search mode must be greater than zero. So here, a zero will be added before the unencrypted plaintext symbol to indicate that it will be only encoded by mask mode and not by search mode.

For the mask mode encryption, it has to extract 8 least significant bits of the chaotic map output X. Since it is a double precision real number of 52 bit size, it has to take 45th bit to 52nd bit for the chaotic mask mode encryption after the Huffman compression. After the chaotic search mode encryption of all the plaintext blocks, the subsequent step is to apply compression technique. The adaptive huffman encoding scheme is here used to compress the output of search mode encryption. A Huffman binary tree is built for all the completed number of iterations, including zero. If there exists more than one mapping table, the same number of repetitions using different mapping tables should correspond to different Huffman codes. When the Huffman binary tree is built, the number of iterations and the special character are replaced by the equivalent variable-length Huffman code to form the intermediate sequence.

An example can illustrate this in detail. Suppose that there are four symbols to encrypt. The first and the fourth plaintext characters are more probable and are encrypted in the search mode. They need five and seven iterations of the chaotic map to land on the target partition and the corresponding Huffman codes denoted as respectively. The second and third plaintext symbols are of lesser probability and are encoded by the mask mode. Therefore, the special character is inserted just after and the second plaintext symbol to specify the mask mode. The length of the intermediary sequence is measured. If it exceeds the length of the plaintext, this means that no compression is achieved at all. In this case, the plaintext sequence should be encoded by the all-mask mode, rather than the hybrid one.

The intermediate sequence should be masked by the binary mask sequence m obtained at the time of iteration before transmission. In this process, both the binary mask sequence and the intermediary sequence are separated into 32-bit blocks. The ciphertext of the block, 'i' is given by equation (4)

$$c = \left(r_{i+1} + m_{c_{i-1}+r_{i+1} mod(L/4)} + c_{i-1}\right) mod 2^{32} \quad (4)$$

Where c_i and r_i are the i^{th} 32-bit block in the ciphertext and intermediate sequences, respectively. The location of the mask block for r_i is determined by c_{i-1}mod (L/4), where L is the plaintext length in bytes. To encrypt the first block c_0, c_{-1} is required, which is calculated by the secret key. If this covering process is performed only once, a tiny change in the last bit of the plaintext sequence may affect only a minor portion at the end of the ciphertext sequence. Therefore, it should be performed at least twice consecutively so as to ensure that a small change in any location of the plaintext sequence spreads over the whole ciphertext sequence. The extra block r_{i+1} required at the end, is set to c_{-1}.

The advantage of this scheme compared with the Baptista method is that it can represent ciphertext with a small number [8]. Because the number of iterations needed for more probable symbols will be less. The chance of getting searched will be high for those symbols which has a number of partitions. In this, the more probable symbols can only be encrypted. The fewer probable symbols will be encrypted by mask mode by a pseudo random bit stream generated from the logistic map.

IV. RSESULT

A. Compression Ratio

The standard files from the Calgary Corpus are used for calculating the compression performance. There are 7 separate files of different types. Two simulation configurations are chosen. The first column, only 16 higher probable plaintext symbols are selected and they are all mapped to one table. In the second column, 128 higher probable plaintext symbols are selected. Then they are mapped to 16 tables of 8 symbols per each. The compression ratio (CR) calculated by equation (5) is listed in Table 4 for different test files.

$$CR = (\text{ciphertext} / \text{Plaintext}) * 100\ \% \quad (5)$$

Table IV shows that all files can be compressed using the two configurations. However, the compression performance of the second configuration (16 maps, eight symbols, each) is better for most of the files.

TABLE IV. COMPRESSION RATIO

Compression Ratio of files		
File	***1 map, 16 symbols***	***16maps, 8symbols each***
Pic	32.93%	31.52%
book1	83.39%	71.15%
Geo	84.20%	85.67%
paper2	84.23%	72.74%
paper3	84.79%	74.51%
book2	85.09%	75.51%
paper4	85.90%	77.31%

B. Plaintext Sensitivity

To test the plaintext sensitivity, a bit is changed at different locations of the plaintext sequence, and then encoded by the same key under the all-mask mode. The two resultant ciphertext sequences are compared bit-by-bit, and the ratio of bit change is calculated. As in Table V, bit change at the beginning of the plaintext calculated as 50.00%, bit change in the middle of the plaintext as 50.04% and at the end of the plaintext as 50.01%. This result shows that the ciphertext is very sensitive to the plaintext even for a bit change at different positions. Two rounds of masking are done and so change spreads over the entire ciphertext sequence. The security of this scheme is high for both the all-mask mode and hybrid mode. For hybrid mode, the ciphertext block length is not fixed due to the occasional insertion of the variable-length Huffman code. It is difficult to identify correctly the bits corresponding to a Huffman code. Moreover, decoding a bit stream of Huffman codes is found difficult without any knowledge about the Huffman coding table. Therefore, the hybrid mode is considered more secure than the all-mask mode.

TABLE V. PLAINTEXT SENSITIVITY

Plaintext Sensitivity Analysis	
Position	***Ratio of Bit change***
Beginning	50.00%
Middle	50.04%
End	50.01%

V. CONCLUSION

Existing approaches for simultaneous chaotic encryption and compression schemes are encryption oriented. Compression is not the main concern, and so, some of them suffer from the low compression performance. On the contrary, algorithm implemented by K.W Wong builds a chaotic cryptosystem that was designed for both encryption and compression. Results indicates that all the standard test files can be compressed to a satisfactory degree, and the ciphertext is very sensitive to a tiny change in the key or the plaintext. Therefore, the compression capability is achieved while the security is maintained. This system also ensures that the ciphertext is not lengthier than the plaintext. Thus it has attained the basic optimization criteria for an encryption algorithm and can be applied in cloud computing. Cloud computing enables an organization to work with minimum infrastructure of resource and data. By giving an online facility to manage its runtime demand of resources and data, cloud computing became popular in its field. By giving a better security feature along with a better compression performance, IT team can exploit it in a good manner. By incorporating Huffman encoding in a search based chaotic cryptosystem, any organization can benefit from an online resource infrastructure with secure data.

REFERENCES

[1] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable secure cloud data storage services," Network, IEEE, vol. 24, no. 4, pp. 19-24, 2010.
[2] C. Wu, and C. Kuo. “Design of integrated multimedia compression and encryption systems.” IEEE Trans. Multimedia, vol. 7 (no. 5), pp. 828–839, 2005.
[3] E. Aguiar, Y. Zhang, and M. Blanton, "An overview of issues and recent developments in cloud computing and storage security." HighPerformance Cloud Auditing and Applications. Springer, pp. 3-33, 2014.
[4] G. Alvarez, and S.Li. “Some basic cryptographic requirements for chaosbased cryptosystems.” Int. J. Bifurcat. Chaos, vol. 16 (no. 8), pp. 2129–2151, 2006.
[5] J. Chen, J. Zhou, and K.W. Wong, “A Modified Chaos-Based Joint Compression and Encryption Scheme.” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 58, (no. 2, p), 2011.
[6] K. W. Wong, “A fast chaotic cryptography scheme with dynamic look-up table.” Phys. Lett. A, vol. 298, pp. 238–242, 2002.
[7] K.W. Wong, and C.H. Yuen, “Embedding compression in chaos-based cryptography,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 55 (no. 11), pp. 1193–1197, 2008.
[8] K.W. Wong, S.W. Ho and C.K Yung, “A chaotic cryptographic scheme for generating short ciphertext.” Phys. Lett. A, vol. 310 (no. 1), pp. 67–73, 2003.
[9] M. S. Baptista, “Cryptography with chaos.” Phys. Lett. A, vol. 240, (no. ½), pp. 50–54, 1998.
[10] M.Y. Javed, and A. Nadeem, “Data Compression Through Adaptive Huffman Coding Scheme.” IEEE, Vol. II, pp.187-190, 2000.
[11] Q. V. Lawande, B. R. Ivan and S.D. Dhodapkar, “Chaos based cryptography: A new approachto secure communications.” BARC News Letter, 2005.
[12] S. Li, G.Chen, K.W. Wong, X. Mou and Y. Cai, “Baptista-type chaotic cryptosystems: Problems and countermeasures.” Phys. Lett. A, vol. 332, pp. 368–375, 2004.
[13] S. Ramgovind, M. M. Eloff, and E. Smith, "The management of security in cloud computing," in Information Security for South Africa (ISSA), [12] 2010. IEEE, pp. 1-7, 2010.