IJCSN International Journal of Computer Science and Network, Volume 4, Issue 2, April 2015
ISSN (Online): 2277-5420 www.IJCSN.org
Impact Factor: 0.417

Steganographic Approach to Ensure Data Storage Security in Cloud Computing Using Huffman Coding (SAHC)

[1] Trijit Chatterjee, [2] Mrinal Kanti Sarkar, [3] Dr. V S Dhaka

[1] Dept.of Computer Science & Engineering, MCKV Institute of Technology, Howrah, India

[2] Dept. of Computer Science & Engineering, University of Engineering & Management Jaipur, Rajasthan, India

[3] Dept. of Computer Science & Engineering, Jaipur National University, Jaipur, India

Abstract - Cloud computing offers the on demand computational infrastructure to the users which has the potential to decrease the huge cost to build IT based services. It can provide ubiquitous, convenient data storage facility. It is a significant issue as the whole data stored to a set of interconnected resource pools which are situated over different location of the world. Stored data can be accessed through virtual machines by unauthorized users. There are different types of security and privacy challenges that are required to analyze and take care. To ensure privacy and security of data in cloud computing, we proposed a new data hiding technique called Steganographic Approach using Huffman Coding (SAHC) which ensures data security in cloud computing during data-at-rest. Our objective is to prevent data access by unauthorized users from cloud storage. The main idea of this paper is to develop a steganographic technique to secure cloud data.

Keywords - *Cloud Computing, Data Storage Security, Huffman Coding, Steganography.*

1. Introduction

Cloud computing is known as a model of latest technology over the internet which satisfy on demand services such as storage, software, resources and network [1]. It is growing rapidly and releasing the new services which required very less management of effort. Users can get the advantage of these various computing services without building its own infrastructure. In fact users can access cloud facility from any computer and from any location of the world. It has the ability to monitor the performance when it is allocate or reallocate the resources dynamically.

Though there are many services that can be provided to the client by the cloud but data store is one of the main features that the cloud service providers provides to the users. But many clients/users are not ready to implement cloud computing model due to the lack of appropriate security mechanism or weakness in protection of data. There are so many cloud computing vendors, such as Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Services (S3), Using Amazon S3 [2], you can store and retrieve huge amount of data from anywhere of the world without any restriction of time and it is just a simple web services interface[2]. This services allow the developer to access the highly secure, scalable, fast, inexpensive reliable, infrastructure.

Data security is an important aspect regarding quality of services. But it is facing various types of security threat for number of reason. So, it is unable to fulfill the quality of services for the several reasons. Firstly we cannot implement the traditional cryptographic technology for the aim of data security as the user' loss their control on data storage. As we don't have the required knowledge of entire data, it is very tough job to verify the actual data using verification strategy. So, we can't implement a verification strategy. This leads to verify correctness of data which are stored in cloud storage and it face more challenging interface. Secondly, it is not a third-party data warehouse where the data will be stored. The data which are stored in cloud storage may be repeatedly modified by the user. So, for this frequent operation, it needs to more advanced technology to avoid data loss from the cloud

storage. Last but not the least, the cloud data storage are running in a cooperated, simultaneously, and in distributed manner [3] and data of every client are stored in multiple physical locations in a random manner. So, we need a robust and protectable data storage technique in the real word which the have distributed protocols for storage correctness and assurance. In this paper, we have proposed a data hiding scheme through Steganographic Approach Using Huffman Coding which ensures explicit dynamic data support and security of data when these data are in the cloud storage. To provide security of data we store the data into several images such that unauthorized users can not view the original content of the data.

The Huffman Tree constructs an optimal prefix code called a Huffman code [4]. The algorithms at first calculate the total number of characters present in the file to be stored in the cloud. The characters are stored in a priority queue along with their frequencies. Let's say, there are six characters A,B,C,D,E and F as shown in Fig-a.

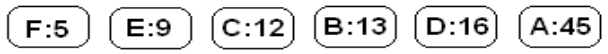

Fig - a

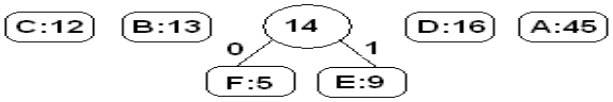

Fig - b

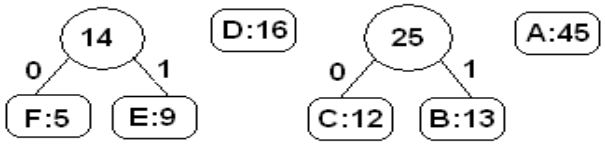

Fig - c

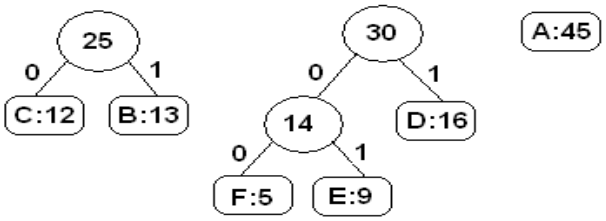

Fig - d

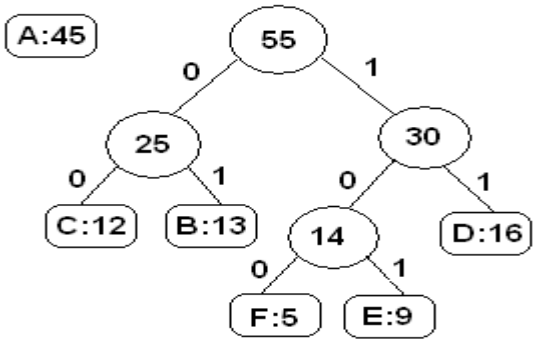

Fig - e

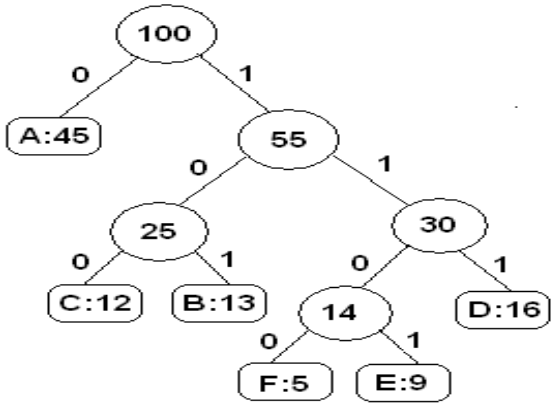

Fig - f

The characters are arranged in ascending order in the priority queue. In each iteration of the algorithm two minimum frequency nodes are taken and they are combined together to form another node whose frequency is the sum of the frequencies of the two nodes that are merged. In figure – b, we can see F and E have minimum frequency. So they are combined to from another node whose frequency is the sum of F's frequency and E's frequency. Each time when a merge operation is done, the left branch of the merged node has the label 0 and the right branch of the merged node has label 1. Figures c-f shows the result of subsequent iterations. Fig-f shows the full Huffman tree. The prefix code of A is 0, B is 101, C is 100, D is 111, E is 1101 and F is 1100. We can see that the character which has highest occurrence or frequency has the minimum prefix code length and lowest frequency has maximum prefix code length. Now for a given code 01001001101 we can decode them to get back the original code by traversing the Huffman tree. On getting the first bit 0 we reach leaf node A, so A is the first character. Again starting from the root of the tree we traverse the label 100 which leads to the leaf node C. Again starting from the root node we will get C. Subsequently prefix code 1101 will lead us to the node F. Thus the characters were 'ACCF'.

Our contributions are summarized on the following aspects:

1) We are storing Huffman codified data into images, which is implicitly compressed by Huffman coding, but if we compare with many of its others works, data are stored in raw format.
2) It prevents data access from unauthorized users from cloud storage, as the malicious users don't know the Huffman codes for the data before hand.
3) Our work incorporates an efficient data storage and retrieval operation using Huffman code.

The paper is organized as follows: we briefly discuss the architecture of cloud computing and its security issues in Section 2. Section 3, we provides our system architecture, design goal, notations and security model. Steganographic approach is presented in Section 4. We analyzed the security and performance evaluations in Section 5. Section 6 provide the related work. Finally, in Section 7, we conclude our remarks on our proposed model and its future scope.

2. Cloud Computing Architecture and Security Issues

Service delivery models and the deployment models are two categories of Cloud computing services [1]. The deployment models are:

1) **Private cloud**: which can be used for single organizations,
2) **Private cloud**: which is provisioned for exclusive use by a particular community of customer,
3) **Public cloud**: is available to public user and they can register and use the available infrastructure, and
4) **Hybrid cloud**: it is a composition of private cloud and public cloud.

Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) are three layers of service delivery model [1].

Software as a Service (SaaS) offers services on demand. It is the topmost layer which provides a complete set of applications. It delivers application for end users and need not require installing application software on the customer's computer.

Platform as a Service (PaaS) provides platform oriented services for software execution. It is the middle layer and it delivers platforms, tools and other business oriented services that enable customer to develop and manage their own application, without installing any of the required platforms.

Infrastructure as a Service (IaaS) supports the basic infrastructure. It is the lowest layer and shares the hardware resources for executing services. It has the right for processing storage, networks etc.

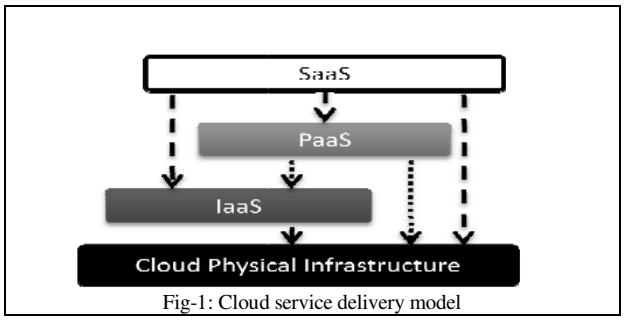

Fig-1: Cloud service delivery model

Fig-1 shows the service delivery model and it has several possible implementations which increase the complexity of the development of standard security model

As we are storing our data and running our software on somebody's CPU with the help of someone else's hard disk So it will face malicious security issues such as phishing, data loss, botnet (Collection of machines are running remotely). Moreover, the multi-tenancy and pooled computing resources have introduced different types of security challenges that need novel techniques to tackle. For example, hackers may use cloud to organize botnet because cloud often offers more reliable infrastructure at a relatively low price for them to start an attack [5].

3. Problem Statement

It is a crucial job to manage data-at-rest plays in cloud computing. The main problem with data-at-rest in cloud is loss of control. An unauthorized user can access the data as data are stored in a shared environment. Though, now-a-days storage devices are empowered by encryption techniques which restrict unauthorized access to data. Encryption methodologies fails to provide authorized access if encryption and decryption keys are available to malicious users. We are providing a stenographic approach which will hide data into images.

3.1 Schematic System Architecture

The architecture of our proposed model [6] is illustrated in Fig-2. In this architecture, we define different network entities which can be identified below.

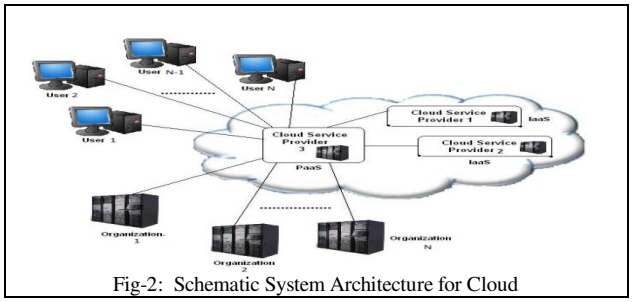

Fig-2: Schematic System Architecture for Cloud

- **User:** Users who want to use cloud infrastructure.
- **Cloud Service Provider-1(CSP-1):** The data will be stored here in the form images.
- **Cloud Service Provider-2(CSP-2):** Encryption and decryption techniques will be stored. This mechanism will hide data into images and retrieve data from images.
- **Cloud Service Provider 3(CSP-3):** All the computations will be taken here by the user. CSP-3 will interact with CSP-1 and CSP-2.

3.2 Security Model

This is the continuation of our last work [6]. We are not storing the data file physically. Instead of storing data into a file, we are storing data into some images. This concept is known as steganography which tells that hide one piece of data/message in such a way that no one except the sender and intended recipient suspects the existence of the data. This is the new paradigm of security through obscurity. For example we divide the entire file into three parts and each part is stored into the corresponding images Fig-3. The division of file is dependent on the size of data file and image.

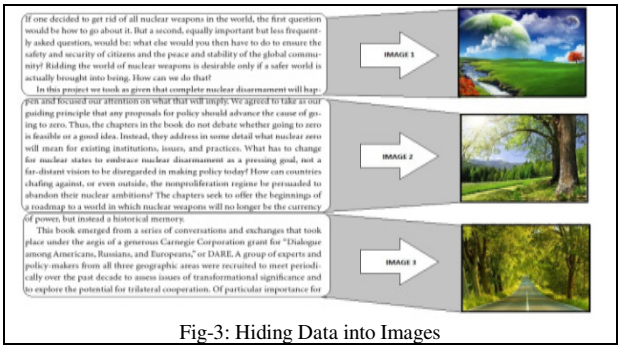

Fig-3: Hiding Data into Images

We present the security model in Fig-4 and Fig-5. The computations done by the users will take place in CSP-3. If a user wants to store their data, the following process will happen:

1. CSP-3 requests CSP-1 for a set of images.
2. CSP-1 will give acknowledge to CSP-3 by sending the required set of images.
3. CSP-3 requests CSP-2 for the data hiding algorithm stored in CSP-2.
4. CSP-2 will send data hiding algorithm to CSP-3 after getting request from CSP-3.
5. CSP-3 applies the steganographic technique for hiding the actual data and data will be saved into the images.
6. Now the images will be sent to CSP-1.

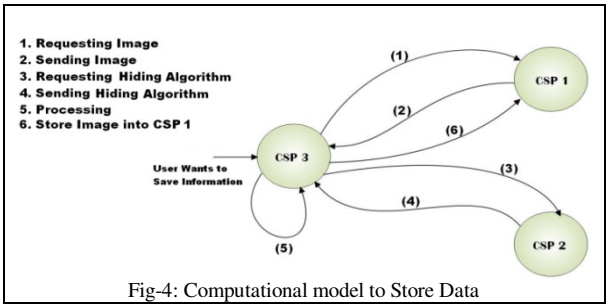

Fig-4: Computational model to Store Data

The following operation will be done whenever the user want to retrieve the data.

1. CSP-3 requests CSP-1 for those set of images which are containing the data.
2. CSP-1 will give acknowledge to CSP-3 by sending the required set of images.
3. CSP-3 requests CSP-2 for the data retrieval algorithm stored in CSP-2.
4. CSP-2 will send data retrieval algorithm to CSP-3 after getting request from CSP-3.
5. Now the CSP-3 applies the retrieval algorithm on the images and those retrieved data will be stored into a file. This file will be displayed to the user.

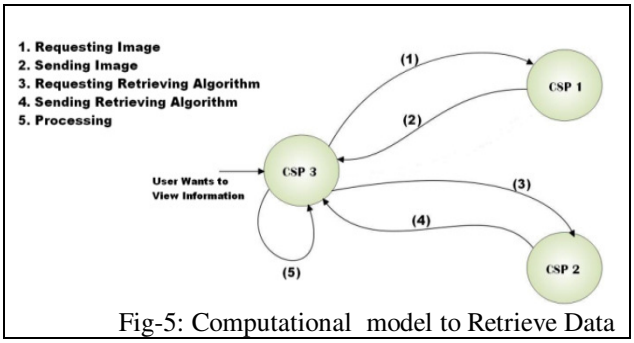

Fig-5: Computational model to Retrieve Data

The temporary file will be deleted when the user want to log out from the system.

3.3 Design Goal

Hiding confidential data into any media is called steganography. In our proposed model we have taken

image as a media. Collection of pixels is needed to form a image. The numerical value determines the color. The range of this numerical value is 0 to 256. The typical digital image is made up of either 8 bit (256 colors) or 24 bit (true color, 8 bits each for red, green and blue) pixels (Picture Elements), where each pixel is composed of 8 bits. As we know that Human Visual System (HVS) has very low sensitivity; it cannot detect small changes in color or patterns. Because of this weakness, we have taken image as a media. As well as variable length encoding does not help the attacker to recognize the characters. When any attacker wants to track images, he/she has no idea about the frequency of each character.

If he/she can't generate the frequency file, he/she also can't generate the Huffman code. So, he/she will also not understand the binary coded data which has sent in the least significant bits of the pixels because; the decoding of bits can only be done by the Huffman tree only. The frequency file contains only frequency, but the frequency of which character is completely unknown since no character is present there. Only values are present, but those values correspond to which character is completely unknown to the attacker. If we change the chronological order, tracking characters will be completely difficult and ultimately we are having a secured system.

- Correctness: Data of file will be stored correctly into images.
- Availability: The data can be retrieved from images by an authorized user when it is required.
- Protection: Human Visual System (HVS) could not detect the original data using an image.

3.4 Notation and Preliminaries

- **F_{Text}:** This is a text file which contents the actual data and we have to secure this data. Initially it will be created by an authorized user and finally it will be store in CSP-1 in the form images. The file will be de-allocated when the all the computation will be done.
- **F_{Tem}:** It is a temporary file which will be used during data processing time (i.e retrieval operation and will be free)
- **F_{Freq}**: It consists only one frequency count which are arranged in some chronological order like at first A to Z then a to z then 0 to 9 then all special characters.
- **C_N:** Number of characters present in a file in F_{Text}.
- **F_{Code}:** It stores codes generated by the Huffman Tree and it will be stored in some chronological order like at first A to Z then a to z then 0 to 9 then all special characters. Only one code will be present in one line.
- **P_{Count}:** Total no. of pixel present in an image.
- **F_{Name}:** When the user wants to retrieve their data from cloud storage, at that time it will be crated
- **Image_Index:** It holds the indexes of valid images.
- **Image_Index1:** A temporary variable which holds the address of image.
- **Image_Database:** It is a database of images which is available in CSP-1, It consists the following attributes of an images:
  a. Name of image
  b. Valid bit, represents the availability of images.
  c. Address field, contain the address of different images and these images have the consecutive data.
- **F_Database:** It maintains the records of set of images where we will store data and it is available in CSP-1. It does not contain the actual information. It actually maintains following attributes:
  a. Name of file
  b. C_N of file.
  c. Address field, location of images.
- **ImageSearch():** It finds a valid image which will be able to store data.
- **MdfImg():** It transfer data into images.
- **RdfImg():**It retrieve original data from an image and store the data into a file.
- **Q:** A priority queue which stores the characters according to their frequencies starting from lowest to highest.

4. Steganographic Approach Using Huffman Coding To Ensure Data Storage Security

In cloud computing, we cannot process the data locally. The data will process remotely, so the security of data must be guaranteed and distributed server is needed. The major concern regarding security of data in cloud computing is that, as data are available in remote servers in raw format. So, it can be easily accessible and can be manipulate by unauthorized users. So, our main aim is to ensure data security in cloud storage such that it can't be detect by any malicious users. We have implemented the following methodologies.

4.1 Image Database

We have created a image database. This database contains a set of images of different sizes which is stored in CSP-1. A set of images will be sent to CSP-3 when a user wants to store data into cloud.

4.2 File Database

File database does not contain the actual information. It is a file and this file holds the address of images where we will store our data.

4.3 Embedded Data into Images

In this section we deal with the files which are to be stored in cloud data storage. It counts the total no of character presents in the file and find the frequency of the occurrences of each character and codifies the original characters by Huffman codes. After that steganography is applied to both frequency of characters and the codified data.

Algorithm 1: HF-Codification ()

1. **procedure**
2. Read file $\mathbf{F_{Text}}$ which is to be saved in Cloud
3. Compute $\mathbf{C_N}$ from $\mathbf{F_{Text}}$
4. Find the frequency of occurrences of each characters in $\mathbf{F_{Text}}$ and store them in some chronological order
5. Store frequency in a new file $\mathbf{F_{Freq}}$.
6. $\mathbf{F_N}$ = Freq-Codification( )
7. Call Huffman-Tree()
8. Create a file $\mathbf{F_{Code}}$
9. Open $\mathbf{F_N}$. Reach **EOF** of $\mathbf{F_N}$ where the original characters of F_{Text} will be replaced by the Huffman codes present in $\mathbf{F_{Code}}$.
10. Calculate the total Bit $\mathbf{B_{Count}}$ in $\mathbf{F_N}$.
11. Delete $\mathbf{F_{Text}}$, $\mathbf{F_{Freq}}$ and $\mathbf{F_{Code}}$.
12. Call Steganography() to perform steganography on $\mathbf{F_N}$
13. **end procedure**

4.4 File Codifications

Here the frequency file is read digit by digit and each digit is codified into 4-bit binary pattern.

Algorithm 2: Freq-Codification ()

1. **procedure**
2. Open $\mathbf{F_{Freq}}$ and a new File $\mathbf{F_N}$.
3. **while** (Read characters from $\mathbf{F_{Freq}}$ until **EOF**)
4. **do if** (character is a new line character)
5. Append 1111 at the end of $\mathbf{F_N}$.
6. **else**
7. Convert the digit to its 4-bit binary form.
8. Append those 4-bits at the end of $\mathbf{F_N}$.
9. **end if**
10. **end while**
11. Append 11111111 at the end of $\mathbf{F_N}$.
12. Return $\mathbf{F_N}$
13. **end procedure**

4.5 Hiding Data within Images Steganography

This algorithm deals the pre-requisite requirements like load image, store file name, image index and finally it call the MdfImg operation which will map the data from a file to an images.

Algorithm 3: Steganography()

1. **procedure**
2. Load Image_Index = ImageSearch (Image_Database)
3. Store (F_{Name}, B_{Count}, Image_Index)
4. MdfImg (Image_Database [Image_Index]);
5. **end procedure.**

4.6 Searching of Valid Image

The algorithm searches an image which we can be used to store the data. It returns the address of a valid image if it is available in image database.

Algorithm 4: ImageSearch(Image_Database)

1. **procedure**
2. Open Image_Database;
3. **for** Image_Database$^{(i)}$, i←1, n **do**
4. **if** (Image_Database$^{(i)}$.valid==1)
5. **return** i
6. **end if**
7. **end for**
8. **end procedure**

4.7 Mapping Data from a File to Image

It does the actual steganographic operation by storing data into images.

Algotithm 5: MdfImg (Image_Database [Image_Index])

1. **procedure**
2. Read Image_Database [Image_Index];
3. Compute **Pixel** $_{\mathbf{Count}}$ for Image_Database[Image_Index];
4. Open $\mathbf{F_N}$
5. **while** (Read Characters until **EOF**)
6. **do if** (**Pixel** $_{\mathbf{Count}}$ < **B** $_{\mathbf{Count}}$)

7. Last bit of each consecutive pixels of the Image_Database[Image_Index] is replaced by Store each character.
8. **else**
9. Load Image_Index1=ImageSearch (Image_Database)
10. Image_Database [Image_Index1].valid=Image_Index1
11. Image_Database [Image_Index 1].valid=0
12. **end if**
13. **end while**
14. **end procedure**

4.8 Retrieving Data from Image

The following algorithm retrieves the data from the images which is kept in cloud storage.

Algorithm 6: RetrieveData ()

1. **procedure**
2. Read **File_Database**;
3. **for** F_Database (i), i=1 to m
4. **do** **if** (F exits)
5. I=Holds the address of image.
6. **end if**
7. **end for**
8. Open Image_Database;
9. Read Image_Database [I];
10. Open a $\mathbf{F}_{\mathbf{Temp}}$ and a $\mathbf{F}_{\mathbf{Freq}}$
11. **while** (Until we get 11111111 in Image_Database [I])
12. **do** Read 4 bits at a time from 4 consecutive pixels
13. Convert them into decimal form.
14. sum =sum + 4
15. **if** (decimal number is within 0 to 9)
16. Write that digit in $\mathbf{F}_{\mathbf{Freq}}$
17. **else**
18. Write new line character in $\mathbf{F}_{\mathbf{Freq}}$
19. **end if**
20. **end while**
21. sum= sum - $\mathbf{B}_{\mathbf{Count}}$
22. Call Huffman-Tree( ) based on the frequency counts present in $\mathbf{F}_{\mathbf{Freq}}$ and create the HuffmanTree.
23. **while** (sum <= $\mathbf{B}_{\mathbf{Count}}$) **do**
24. read bits from Image_Database [I]
25. Start traversing the Huffman-Tree from root.
26. When we reach leaf node, we will get character.
27. Append that character in $\mathbf{F}_{\mathbf{Temp}}$.
28. Increment sum number of times we collect bits from Image_Database [I]
29. **end while**
30. Show $\mathbf{F}_{\mathbf{Temp}}$ to the user, after user closes the file $\mathbf{F}_{\mathbf{Temp}}$, delete the file $\mathbf{F}_{\mathbf{Temp}}$ from system.
31. **end procedure**

4.9 Construction of Huffman Tree

Here, we are presenting the generation of Huffman Tree. Given a character X with frequency distribution { f(x) : x € X}. A priority queue, Q, of nodes, is used to generate Huffman Tree with levels (frequency) as key

---

Algorithm 7a: Huffman-Tree (X)

1. **procedure**
2. $\mathbf{F_N}$=**|X|**
3. **Q=X**
4. **for** i=1 to **N**-1
5. **do**
6. Z=Allocate_node( )
7. Z.left=Extract_min(Q)
8. Z.right=Extract_min(Q)
9. Frequency(Z)=Frequency(Z.left)+Frequency(Z.right)
10. Insert(Q,Z)
11. **end for**
12. **end procedure**

Algorithm 7b: Allocate_node()

1. **Procedure**
2. Create a node for storing characters and their frequency from available free memory space.
3. Return the allocated node.
4. **End procedure**

Algorithm 7c: Extract_min(Q)

1. **procedure**
2. Remove and return the character with minimum frequency from the priority queue **Q**.
3. **end procedure**

Algorithm 7d: Insert(Q,Z)

1. **procedure**
2. Insert the node Z in the priority queue Q
3. **end procedure**

---

5. Security Analysis and Performance Evaluation

The security and the efficiency of the proposed model SAHC depends on our proposed system architecture and its security model which is defined in section II. We evaluate the performance of our model to implement the image database file database.

5.1 Security Strength against HF Codification

Huffman coding is a variable length coding scheme where the most occurring character has the least code length and the least occurring character has the highest code length. The frequency of each character is stored in some chronological order like at first A to Z then a to z then digits and separators etc. The frequency file contains each frequency counts in each line. Variable length encoding does not help the attacker to recognize the characters. When any attacker wants to track images, he/she has no idea about the frequency of each character. If he/she can't generate the frequency file, he/she also can't generate the Huffman code. So, he/she will also not understand the binary coded data which has sent in the least significant bits of the pixels because; the decoding of bits can only be done by the Huffman tree only.

The frequency file contains only frequency, but the frequency of which character is completely unknown since no character is present there. Only values are present, but those values correspond to which character is completely unknown to the attacker. If we change the chronological order, tracking characters will be completely difficult and ultimately we are having a secured system.

5.2 Security Strength against CSP-1

CSP-1 only stores some files. These file contain the location of images. It also has the set of images which has the data. But an unauthorized user cannot get anything to seeing these images because of HVS. CSP-1 does not contain the retrieving algorithm, thus the images containing data are purely safe.

5.3 Security Strength against CSP-2

Retrieving and hiding mechanism are stored in CSP-2, these will be needed at the time of viewing and storing the data from CSP-3. CSP-2 does not contain the images, thus knowing only the algorithm will not help the attacker.

5.4 Security Strength against CSP-3

In our proposed model, CSP-3 is responsible for computation i.e. store data into images and retrieve data from images. All the files will be deleted after the above operations. So, there is nothing to fear from data loss and unauthorized access.

6. Related Work

Cong Wang et al. [3] use homomorphic token with distributed verification of erasure-coded data. It ensures data storage security and finds the location the server which has been attacked. It support update, delete and append operation on data blocks such. But it is fail to achieve public verifiability and storage correctness. Shacham et al. [7] build a random linear function using homomorphic authenticator which is useful for unlimited number of queries without taking of communication overhead. Jules et al [8] proposed Proofs of Retrievability (POR) for large files. Later Bowers et al. [9] improved POR protocols which generalize both Juels and Shacham's work. In their subsequent work, Bowers et al. [10] used distributed systems to extend the POR model.

Shantanu pal et al. [11] ensures to find location of adversary or the attacking party from its target. It is ensuring a more secure platform for the other virtual machine. It may try to attack them, if adversary knows the location of the other VMs. This may harm the other VMs in between. Flavio Lombardi et al. [12] proposed to check the behavior of cloud resources. Executable system file can be monitored by logging and periodic checking. But it encountered faces system performance. Shah et al. [13] proposed to keep the record of online storage a TPA. It encrypts the data using symmetric-keyed hashes, and then it will be send to the auditor computed. However, this approach is applicable on encrypted files and long-term state is maintained by auditors. Schwarz et al. [14] used file integrity across multiple distributed servers to ensure security using erasure-coding and block-level file integrity checks.

Ateniese et al. [15] proposed the "provable data possession" (PDP) model to ensure possession of file in untrusted storages. This scheme used public key based homomorphic tags to audit the data file and it is providing public verifiability. In their subsequent work, Ateniese et al. [16] proposed a PDP scheme that uses only symmetric key cryptography. This method has only lower-overhead than their previous works. The modification (i.e block updates, deletions and appends to the stored file) of stored

file can be done. But this scheme can be used on single server and it is not able to rectify small data corruptions. Distributed scenario and data error recovery issue is unexplored. Curtmola et al. [17] proposed to ensure data possession multiple across the distributed system. It extended PDP which ensure data possession on multiple replicas without encoding each replica separately. It guarantees that multiple copies of data are actually maintained.

7. Conclusion

The main aim of this paper is to present an overview of security challenges that are happening in cloud data storage and introduce a new combined encrypting and data hiding technique to prevent unauthorized data access in cloud data storage. We applied steganographic approach to ensure data storage security in cloud computing using Huffman Coding (SAHC). It is an efficient steganographic strategy for enhancing security of cloud data when it is at rest. Simply we stored the data into images stored in the cloud data storage. Unauthorized users can not rectify the original content of the data due to HVS. Through detailed security and performance analysis we show that our approach gives high security of data when it is on rest in the data center of any Cloud Service Provider (CSP).This proposed architecture will be able to provide customer satisfaction to a great level and it will attract more clients in the field of cloud computing for industrial as well as future research firms.

References

[1] Peter Mell, Timothy Grance, "The NIST Definatin of Cloud Computing", Jan, 2011.http://docs. ismgcorp. com/files/external/Draft-SP-800-145_cloud-definition.pdf.

[2] Amazon.com, "Amazon Web Services (AWS)", Online at hppt://aws.amazon.com, 2008.

[3] Con Wang, Qian Wang, Kui Ren, and Wenjng Lou, "Ensuring Data Storage Security in Cloud Computing",17th International workshop on Quality of service, USA, pp1-9, 2009, IBSN:978-42443875-4.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Third Edition, Prentice Hall of India, 2010.

[5] B.P Rimal, Choi Eunmi,I.Lumb, "A Taxonomy and Survey of Cloud Computing Sytem", Intl. Joint Conference on INC, IMS and IDC, 2009,pp.44-51, Seoul,Aug, 2009. DOI : 10.1109/NCM.2009.218.

[6] M. K Sarkar and T. Chatterjee, "Enhancing Data Storage Security in Cloud Computing Through Steganography", ACEEE International Journal on Network Security, ISSN: 2152-5064, Vol. 5, Issue. 1, pp: 13-19, Jan 2014.

[7] H. Shacham and B. Waters, "Compact Proofs of Retrievability", Proc. of Asiacrypt '08, Dec. 2008.

[8] A. Juels and J. Burton S. Kaliski, "PORs: Proofs of Retrievability for Large Files", Proc. of CCS '07, pp. 584–597, 2007.

[9] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage", Cryptology ePrint Archive, Report 2008/489, 2008, http://eprint.iacr.org/.

[10] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage", Cryptology ePrint Archive, Report 2008/489, 2008, http://eprint.iacr.org/.

[11] Shantanu Pal, Sunirmal Khatua, Nabendu Chaki, Sugata Sanyal, "A New Trusted and Collaborative Agent Based Approach for Ensuring Cloud Security", Annals of Faculty Engineering Hunedoara International Journal of Engineering (Archived copy), scheduled for publication in vol. 10, issue 1, January 2012. ISSN: 1584-2665.

[12] Flavio Lombardi, Roberto Di Pietro, "Secure Virtualization for Cloud Computing ", Journal of Network and Computer Application, vol. 34, issue 4, pp 1113-1122, July 2011, Academic Press td London, UK.

[13] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to Keep Online Storage Services Honest", Proc. 11th USENIX Workshop on Hot Topics in Operating Systems (HOTOS '07), pp. 1–6, 2007.

[14] S. J. Schwarz and E. L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage", Proc. of ICDCS '06, pp. 12–12, 2006.

[15] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores", Proc. ofCCS '07, pp. 598–609, 2007.

[16] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession", Proc. of SecureComm '08, pp. 1–10, 2008.

[17] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession", Proc. of ICDCS '08, pp. 411–420, 2008.

Trijit Chatterjee received the B.Sc. degree in Computer Science (Hons.) from Calcutta University in 2010, M.Sc. degree in Computer Science from St. Xavier's College Kolkata in 2012. Currenlty he is doing his M.Tech. from MCKV Institute of Technology, Howrah, and also working as Teaching Assistant in MCKV Institute of Technology, Howrah, India. His current research interest includes Security in Cloud Computing and Image Processing.

Mrinal Kanti Sarkar received the B.Tech degree in Computer Science & Engineering from Govt. College of Engineering & Ceramic Technology in 2008, M.Tech degree in Computer Science & Engineering from Indian Institute of Technology, Kharagpur in 2010. Currenlty he is doing his PhD. from Jaipur National University Jaipur. He served as a Senior Lecturer in The ICFAI University Tripura from 2010 to 2012. Now, he is working as Assistant Professor in University of Engineering and Management, Jaipur, India. His current research interest includes Security in Cloud Computing, Distributed System and Information Security.

Dr. V.S. Dhaka received his M.Tech. and Ph.D. in Computer Science from Dr. B.R. Ambedkar University, Agra, India. Currently he is working as a Professor in Department of Computer Science & Engineering, Jaipur National National University, Jaipur, India. He is is also Head, Dept. of Computer Science & Engineering, JNU Jaipur.He has 11 years of experience in the industry and academics. He has more than 32 publications in international journals. He always strives to achieve academic excellence. His current research interest includes Neural Network, Information Security, and Image Processing.