Renovation in Compression Expertness of Huffman Coding and Intelligent Data Encryption

Farrukh Fareed
Assistant Professor, RIMT
Bareilly, India

Shiva Chaudhary
Assistant Professor, RIMT
Bareilly, India

ABSTRACT

Data compression is used for reducing storage requirements. It involves transforming data of a given format, called source message, to data of a smaller sized format, called compressed message. The Data compression helps in reducing the size of the database so that it is very easy to maintain huge database. On the other hand Type Casting is used to convert the input source message into suitable format so that Huffman Algorithm can work on maximum data formats and after that Data Normalization technique is used to remove redundancy from the data this technique enhance the compression efficiency of Huffman Algorithm. Decreasing the amount of data required representing a source of information while preserving the original content as much as possible and after that providing security to this compressed code word with the application of data encryption. The main objectives of this paper are to get higher compression efficiency by applying a planned technique on Huffman Algorithm and providing security to this compressed data to limit the unauthorized access using data encryption.

Keywords

Type Casting, Data Normalization, Huffman Algorithm, Data Encryption.

1. INTRODUCTION

A compression algorithm can be used for minimizing the storage space of data. Development of Information technology leads increment in data size so that data compression algorithm is necessary to maintain huge amount of data economically. There are basically two types of compression techniques - lossless and lossy compression depends upon data type, such that in text compression every character of text is important and loss of single character can change the meaning of the sentence, only lossless compression can give desirable results. Compression and Decompression is performed in two parts. In the first part an encoding algorithm generates the compressed code word of input message and in the second part a decoding algorithm that reconstructs the original input message from the compressed code word when ever needed. It is also important to consider the security aspects of the data being compressed so as to limit the unauthorized access to the private and confidential data. In this paper, the proposed technique enhanced the compression ratio and compression efficiency of the Huffman Coding on text data wit an added security using data encryption. This paper also outlines the use of Typecasting which makes Huffman algorithm applicable on more data formats and after that Data Normalization is used for improving compression efficiency as it removes redundancy from data.

2. TYPE CASTING

Typecasting is a technique which is used for changing the contents of one data type into another. Typecasting is used to take benefit of some important features of type hierarchies or type representations. The example of typecasting would be small integers, which can be stored in a compressed format and converted to a larger representation when used in arithmetic computations. In object-oriented programming with the use of typecasting programmers can treat objects of one type as one of their predecessor types to simply interact with them. Different programming language has different rules through which data types are converted. Generally objects and original data types are converted it depends upon requirement. The basic example would be an equation which has both integer and floating point numbers i.e. 3-0.1 in which the integers are usually converted into the latter. Unambiguously the type conversions can either be performed through some special syntax or through separately defined conversion rules. In programming languages *conversion* and *casting* are the two different concepts. Languages mention *conversion*, changing a value from one data type to another. On the other hand the word *cast* refers to clearly changing the *understanding* of the *bit pattern* representing an entity from one type to another. The use of Typecasting before Huffman Algorithm is very useful so that Huffman Algorithm can be applicable to more data formats as compared to conventional Huffman Algorithm.

3. DATA NORMALIZATION

The method of organizing data to decrease redundancy and anomalies is called normalization. Redundant data wastes disk space and creates maintenance problems. The aim of database normalization is to decompose relations with anomalies in order to produce smaller, logical relations. Normalization frequently involves separating large, badly-formed tables into smaller, well-formed tables and defining links between them. Main objective of data normalization is to separate data so that modifications of a field can be made in just one table and then propagated through the rest of the database using the defined relationships. Due to these properties of normalization technique higher compression ratios in Huffman algorithm can be achieved as the data is being normalized before compression. There are certain set of rules for database normalization. Each set of rules are called a "Normal Form." If the first set of rules is applied, the database is said to be in "First Normal Form (1NF)."If the second set of rules is applied on the data which is in first normal form, database is said to be in "Second Normal Form (2NF)" and so on. Although other levels of normalization are also possible, Third Normal Form (3NF) is to be considered the highest level essential for most applications.

3.1 First Normal Form (1NF)

The First normal form (1NF) is defined as belongings of a relation in a relational database.

- ➢ The table cells must be of single value.
- ➢ Repeating groups must be excluded from the Individual tables.
- ➢ A separate table should be formed for each group of linked data.
- ➢ Every group of linked data should be identified by a primary key.

If a database satisfies above set of rules then it is in first normal form.

3.2 Second Normal Form (2NF)

The database is in second normal form if it satisfies the following set of rules:

- ➢ Database should be in First Normal Form (1NF).
- ➢ All non-key attributes should be completely functional dependent on the primary key.

There should be no partial dependencies if partial dependencies occur on primary key it should be removed. There is still a probability for a table in this form (2NF) to reveal transitive dependency.

3.3 Third Normal Form (3NF)

A database is in third normal form if it satisfies the following set of rules:

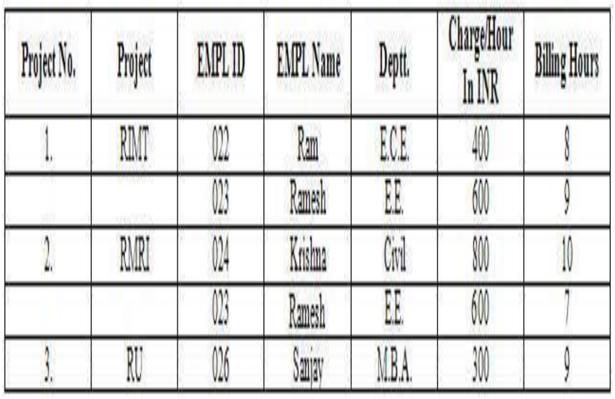
- ➢ Database should be in Second Normal Form (2NF).
- ➢ Remove such domains that are not dependent on the primary key.
- ➢ Remove transitive functional dependency.

Transitive functional dependency is defined via the following relationships: A is functionally dependent on B, and B is functionally dependent on C. In such situation, C is transitively dependent on A via B.

3.4 Example of Data Normalization

Consider a database of Project Report Layout as shown in Table1.

Table 1. Un-Normalized database

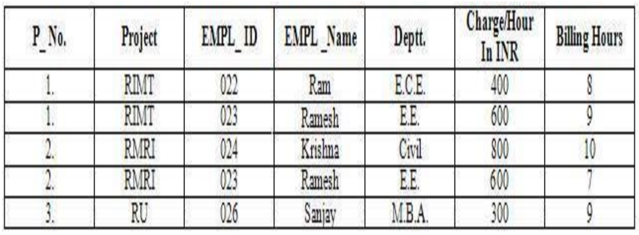


Project No.	Project	EMPL ID	EMPL Name	Deptt.	Charge/Hour In INR	Billing Hours
1.	RIMT	022	Ram	E.C.E.	400	8
		023	Ramesh	E.E.	600	9
2.	RMRI	024	Krishna	Civil	800	10
		023	Ramesh	E.E.	600	7
3.	RU	026	Sanjay	M.B.A.	300	9

3.4.1 First Normal Form (1NF)

Repeating groups must be excluded. Every attribute is dependent on primary key and all attributes are clearly defined. Table 2 has partial dependency.

Table 2. Database is in 1NF

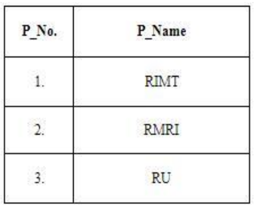


P_No.	Project	EMPL_ID	EMPL_Name	Deptt.	Charge/Hour In INR	Billing Hours
1.	RIMT	022	Ram	E.C.E.	400	8
1.	RIMT	023	Ramesh	E.E.	600	9
2.	RMRI	024	Krishna	Civil	800	10
2.	RMRI	023	Ramesh	E.E.	600	7
3.	RU	026	Sanjay	M.B.A.	300	9

3.4.2 Second Normal Form (2NF)

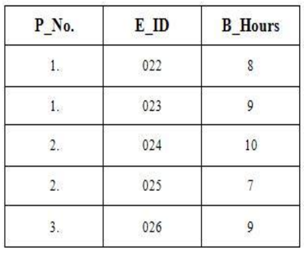
According to the set of rules of 2NF redundant data should be removed. Table 5. Shows transitive dependency.

Table 3. Project



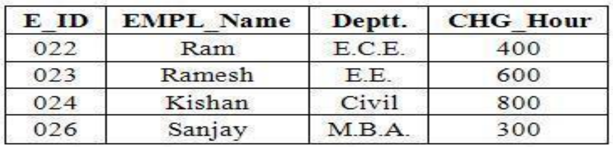
P_No.	P_Name
1.	RIMT
2.	RMRI
3.	RU

Table 4. ASSIGN



P_No.	E_ID	B_Hours
1.	022	8
1.	023	9
2.	024	10
2.	025	7
3.	026	9

Table 5. Employee Details

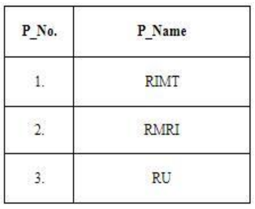


E_ID	EMPL_Name	Deptt.	CHG_Hour
022	Ram	E.C.E.	400
023	Ramesh	E.E.	600
024	Kishan	Civil	800
026	Sanjay	M.B.A.	300

3.4.3 Third Normal Form (3NF)

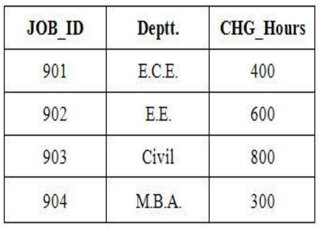
According to the set of rules 3NF, remove columns that are not dependent on key. Remove transitive dependency.

Table 6. Project



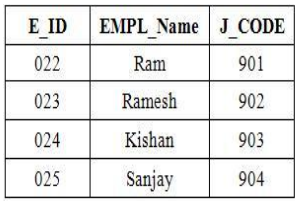
P_No.	P_Name
1.	RIMT
2.	RMRI
3.	RU

Table 7. Job details

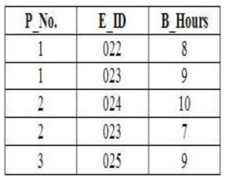


JOB_ID	Deptt.	CHG_Hours
901	E.C.E.	400
902	E.E.	600
903	Civil	800
904	M.B.A.	300

Table 8. Employee Details
Table 9. ASSIGN



E_ID	EMPL_Name	J_CODE
022	Ram	901
023	Ramesh	902
024	Kishan	903
025	Sanjay	904



P_No.	E_ID	B_Hours
1	022	8
1	023	9
2	024	10
2	023	7
3	025	9

Applying data normalization before Huffman Algorithm is extremely advantageous for higher compression ratio and compression efficiency.

4. HUFFMAN ALGORITHM

Huffman algorithm is a lossless text compression algorithm. With the use of Huffman algorithm compression of text data can be done in such an efficient way that every character of text can be recovered when decompression algorithm is employed and this is one of the major factor to use Huffman algorithm in this paper. Huffman algorithm constructs a binary tree to minimize the average bit rate and this binary tree is known as optimal prefix tree. Huffman algorithm gives variable length code as every character in a data file are converted to a binary code in such a way that the high frequency characters in the file have the shortest binary codes and the lowest frequency have the longest binary codes.

4.1 Huffman Data Compression Algorithm

To view how Huffman compression algorithm works, consider an example: **"ABAEACABDCABA"**

1. First step, arrange the characters from highest to lowest frequency of occurrence as follows:

Character	**Frequency**
A	6
B	3
C `	2
D	1
E	1

2. Second step, the two lowest-frequency characters are selected, logically grouped together and add their frequencies. In this example, the D and E characters have a combined frequency of 2 and connection to a parent node which makes it easy to read the code in reverse starting from a leaf node. Interior nodes contain symbol frequencies, connection to two child nodes and the possible connection to a parent node. As a frequent rule, bit '0' represents the left child and bit '1'represents following the right child

Character	**Frequency**
A	6
B	3
C	2
1 (D+E)	2

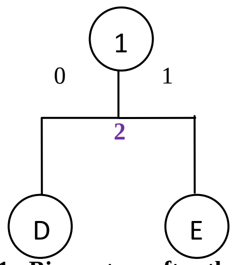


Fig 1: Binary tree after the IInd step

3. Now again select the two elements of the lowest frequencies and make the table and tree according to the rule followed in step 2.

Character	**Frequency**
A	6
B	3
2(D+E+C)	4

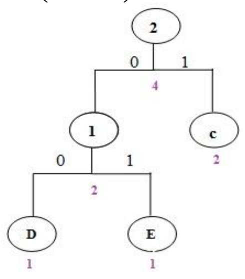


Fig 2: Binary tree after the IIIrd step

4. Now again select the two elements of the lowest frequencies and make the table and tree according to the rule followed in step 2.

Character	**Frequency**
A	6
3(D+E+C+B)	7

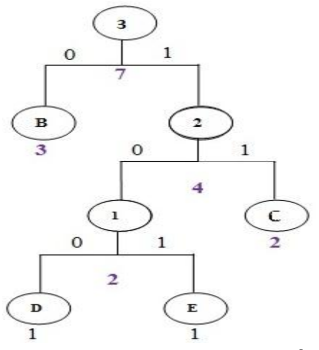


Fig 3: Binary tree after the IVth step

5. Now again select the two elements of the lowest frequencies and make the table and tree according to the rule followed in step 2.

Character	**Frequency**	**Character**	**Frequency**
3(D+E+C+B)	7		
A	6 ⟶	4(A+ D+E+C+B)	13

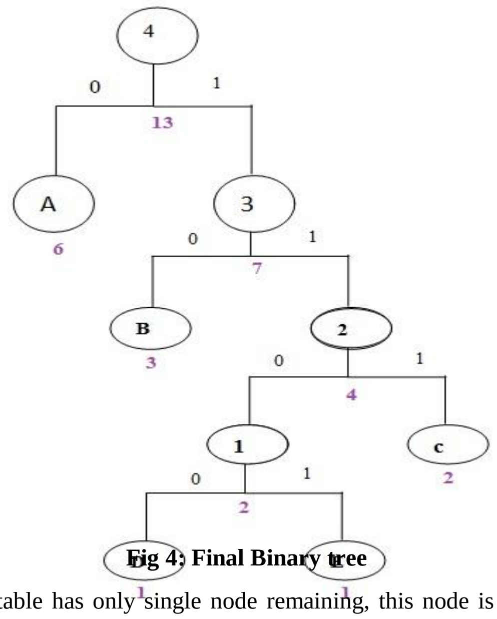


Fig 4: Final Binary tree

When the table has only single node remaining, this node is the root of the Huffman binary tree. The code word for corresponding symbol can be defined from the way from root node to the leaf node:

Table 9. Binary Code Table

Character	Frequency	Code
A	6	0
B	3	10
C	2	111
D	1	1100
E	1	1101

Input message:
ABAEACABDCABA
Compressed code by Huffman Compression Algorithm:
01001101011101011001110100

4.2 Huffman Data Decompression Algorithm

Decompression algorithm is basically converting the prefix codes in corresponding byte values. Huffman decompression Algorithm starts from the root of binary tree and whenever a leaf node reached stop there and look the code of that corresponding symbol from root node to leaf node, assign the corresponding symbol of that code from the compressed code repeat this process until whole compressed message is decoded:

01001101011101011001110100

Code word	Character
0	A
10	B
0	A
1101	E
0	A
111	C
0	A
10	B
1100	D
111	C
0	A
10	B
0	A

Compressed code word:
01001101011101011001110100
Input message in its original form by Huffman decompression Algorithm:
ABAEACABDCABA

5. DATA ENCRYPTION

Encryption can be defined as to make the data unreadable for everybody else except those authorized users. Data Encryption can also be understood as writing the text in such a secret form with certain set of rules such that those who know the set of rules can read the text so it is some form of secret writing. This secret writing is known as Cryptography. Encryption is done by applying a mathematical function to the text and converting it to cipher text. Data Encryption is the most economical and most widely used technique for providing security to sensitive or confidential data. The most difficult part in encryption is to make sure that the authorized users can be able to decipher the text. There are two techniques of encryption Private Key Encryption and Public Key Encryption.

5.1 Private Key Encryption

The Private Key encryption is also known as symmetric key encryption. It is the most conventional method of encryption. In private key encryption single secret key is used for both encryption and decryption process. In this encryption process the secret key must be shared between sender and receiver in a very secure manner. If someone hacks this secret key and calculate the algorithm then the data can be decrypted easily so in this form of encryption there is a need of secure exchange of secret key and a powerful encryption algorithm.

5.2 Public Key Encryption

The Public Key Encryption is also known as asymmetric key encryption. This is a modern approach in this type of encryption there is a use of two types of keys, private key & public key. Public key is open to all everyone knows but private key is a confidential one, assign only to the authorized users. The encryption is done with the help of public key encryption but decryption can only be performed by the mathematical matching of public key and private key.

6. PROPOSED DATA ENCRYPTION AND DECRYPTION TECHNIQUE

Consider the length of the compressed code word that is generated by Huffman algorithm is L and length of generator polynomial S. The remainder of length S-1 is transmitted to authorized user before the process of encryption. In encryption process the length of the code word (L) in divided with the length of generator polynomial (S) such that the nearest prime number of the quotient is multiplied with the generator polynomial. The result of this multiplication is in the form of bit pattern, used for encryption. Let the length of this bit pattern is N. These N bits are XORed with the last N bits of the code word. The process will be done until the last bit pattern of the code word; if code word length is less than N they will be left as it is. As a result of this process the cipher text is generated.

When the authorized user gets this cipher text, the user will be able to do decryption is reverse steps of encryption along with the remainder which is send before compression as XOR of XOR is the same number.

The proposed encryption technique can be embedded very easily with the compressed code word generated by the Huffman algorithm and save extra processing and provide strong security to the confidential data that’s why the name Intelligent Data encryption is used in this paper.

7. OPERATION SUMMARY

To perform the proposed techniques MATLAB software is used. The whole process is performed in two halves and these two halves can be understood with the help of flow charts.

7.1 First Half

The first half can be understood with the help of flow chart as shown in figure below:

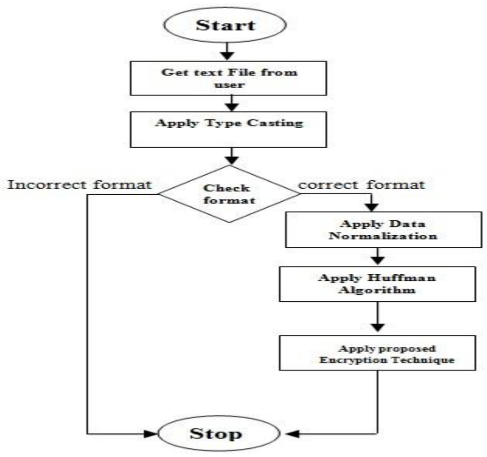


Fig 5: Flow chart of Compression & Encryption

From the diagram the process is done in following main points which are as follows:

- MATLAB module takes the normal text file from the user.
- Apply Type Casting to that file.
- Check format if it is in suitable format than apply next step otherwise stop.
- Apply Data Normalization to remove redundancy.
- Apply Huffman Compression Algorithm to compress the text file and get the compressed Code word.
- Apply proposed encryption technique to get the Compressed and encrypted file.

7.2 Second Half

The second half is the reverse process and can be understood with the help of flow chart as shown in figure below:

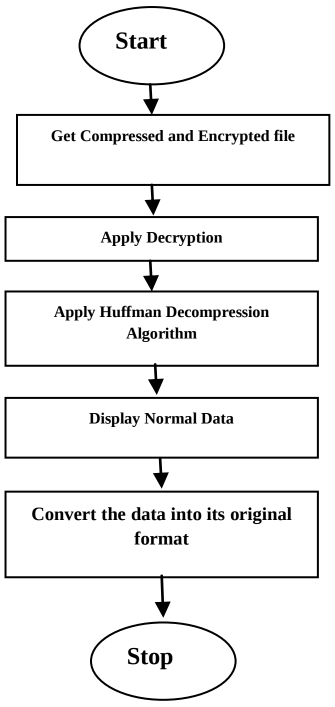

Start
Get Compressed and Encrypted file
Apply Decryption
Apply Huffman Decompression Algorithm
Display Normal Data
Convert the data into its original format
Stop

Fig 6: Flow chart of Decompression & Decryption

From the diagram the process is done in following main points which are as follows:

- Get the Compressed and Encrypted data.
- Apply Decryption to remove security so that authorized user can access the data.
- Apply Huffman Decompression Algorithm to decompress the data.
- Display normal data as operation of normalization is being performed before compression.
- Convert and save the data in its original format so that authorized user can access the data.

8. RESULT

The proposed technique is performed with the help of MATLAB and the summary of the result is shown in the table below and the encrypted output is shown with the help of pictures.

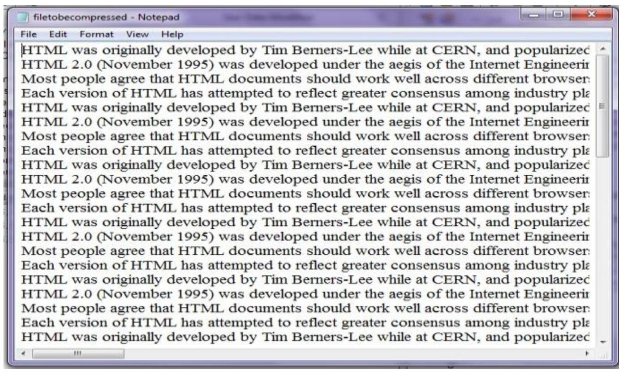
Table 8. Result summary

Size of input text file (kb)	Size of compressed file(kb)	Size of decompress file(kb)	% compression (approx)
22	13	22	40
42	24	42	42
76	43	76	43

Using different-different data in text file and apply proposed techniques, following results obtained as shown in the table above.

Average % compression achieved = $\frac{40+42+43}{3}$ = 41 % (Approx)

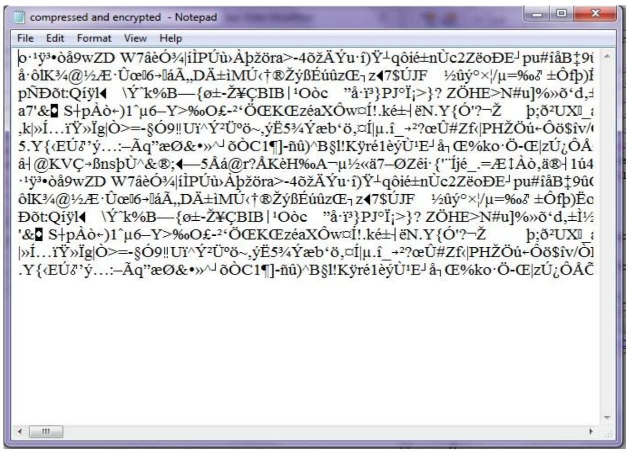
Encryption is also performed along with compression. The picture below shows the text file which is to be compressed and encrypted:



```
HTML was originally developed by Tim Berners-Lee while at CERN, and popularizec
HTML 2.0 (November 1995) was developed under the aegis of the Internet Engineerin
Most people agree that HTML documents should work well across different browser:
Each version of HTML has attempted to reflect greater consensus among industry pla
HTML was originally developed by Tim Berners-Lee while at CERN, and popularizec
HTML 2.0 (November 1995) was developed under the aegis of the Internet Engineerin
Most people agree that HTML documents should work well across different browser:
Each version of HTML has attempted to reflect greater consensus among industry pla
HTML was originally developed by Tim Berners-Lee while at CERN, and popularizec
HTML 2.0 (November 1995) was developed under the aegis of the Internet Engineerin
Most people agree that HTML documents should work well across different browser:
Each version of HTML has attempted to reflect greater consensus among industry pla
HTML was originally developed by Tim Berners-Lee while at CERN, and popularizec
HTML 2.0 (November 1995) was developed under the aegis of the Internet Engineerin
Most people agree that HTML documents should work well across different browser:
Each version of HTML has attempted to reflect greater consensus among industry pla
HTML was originally developed by Tim Berners-Lee while at CERN, and popularizec
HTML 2.0 (November 1995) was developed under the aegis of the Internet Engineerin
Most people agree that HTML documents should work well across different browser:
Each version of HTML has attempted to reflect greater consensus among industry pla
HTML was originally developed by Tim Berners-Lee while at CERN, and popularizec
```

Fig 7: Text File which is to be compressed and encrypted

Picture below shows result of compressed and encrypted file:


compressed and encrypted - Notepad
File Edit Format View Help

```
þ·¹ÿ³•òå9wZD W7âèÓ¾|íÌPÚù›Àþžöra>-4õžÄÝu·î)Ÿ┴qôié±nÙc2ZëoÐE┘pu#îåB‡9í
å·ôlK¾@½Æ·Ûœ6→áÃ„DÄ±iMÚ‹†®ŽýßÉúûzŒ┐z◄7$ÚJF    ½ûý°×¦/µ=‰♂ ±Ôfþ)Ì
pÑÐõt:Qíÿl◄   \Ý˜k%B—{ø±-Ž¥ÇBIB|¹Oòc   ”å·ï³}PJ°Ï¡>}? ZÖHE>N#u]%»õ‘d,±
a7'&◘ S┼pÀò←)1ˆµ6–Y>‰O£-²‘ÖŒKŒzéaXÔw¤Í!.ké±┤ëN.Y{Ó'?¬Ž      þ;ð²UX_
,k|»Í…ïŸ»Ïg|Ò>=-§Ó9‼Uï^Ý²Üºö~,ýË5¾Ýæb‘ö,¤Í|µ.î_→²?œÛ#Zf‹|PHŽÖú←Ôö$îv/
5.Y{‹EÚ♂’ý…:–Ãq”æØ&•»^┘õÒC1¶]-ñû)^B§l!Kÿré1èýÙ¹E┘å┐Œ%ko·Ö-Œ|zÚ¿ÔÅ
â┤@KVÇ→ßnsþÙ^&®;◄—5Åá@r?ÅKèH‰A¬µ½«ä7–ØZêi·{"¨Íjé_.=Æ↕Àò,ä®┤1ú4
·¹ÿ³•òå9wZD W7âèÓ¾|íÌPÚù›Àþžöra>-4õžÄÝu·î)Ÿ┴qôié±nÙc2ZëoÐE┘pu#îåB‡9ûÇ
ôlK¾@½Æ·Ûœ6→áÃ„DÄ±iMÚ‹†®ŽýßÉúûzŒ┐z◄7$ÚJF    ½ûý°×¦/µ=‰♂ ±Ôfþ)Ëo
Ðõt:Qíÿl◄   \Ý˜k%B—{ø±-Ž¥ÇBIB|¹Oòc   ”å·ï³}PJ°Ï¡>}? ZÖHE>N#u]%»õ‘d,±Ì½
'&◘ S┼pÀò←)1ˆµ6–Y>‰O£-²‘ÖŒKŒzéaXÔw¤Í!.ké±┤ëN.Y{Ó'?¬Ž      þ;ð²UX_
|»Í…ïŸ»Ïg|Ò>=-§Ó9‼Uï^Ý²Üºö~,ýË5¾Ýæb‘ö,¤Í|µ.î_→²?œÛ#Zf‹|PHŽÖú←Ôö$îv/Òl
.Y{‹EÚ♂’ý…:–Ãq”æØ&•»^┘õÒC1¶]-ñû)^B§l!Kÿré1èýÙ¹E┘å┐Œ%ko·Ö-Œ|zÚ¿ÔÅC
```

Fig 7: Compressed & Encrypted text file

9. CONCLUSION

This paper experimentally prove that compressed representation of text data lead to significant savings of storage space so that it is very economical and beneficial to maintain huge amount of data base and if encryption is also done after compression this leads to maintain the database secure so that only authorized users can access the database. By using the proposed techniques transmission of huge data over any medium such as via internet, removable storage etc. can be done precisely and there is no need of providing extra security as encryption is already performed. So proposed technique improves the compression expertness of Huffman Algorithm with the use of type casting and Data Normalization before Huffman Algorithm and with the use proposed encryption technique security is provided to the compressed file so as to prevent unauthorized access.

10. REFERENCES

[1] Mohd. Faisal Muqtida and Raju Singh Kushwaha, "Improvement in Compression Efficiency of Huffman Coding”, International Journal of Computer Applications (0975 – 8887) Volume 45– No.24, May 2012.

[2] S.Mohankrishna, Singuru SriHari, T.V. Trinadh, G. Raja Kumar,"A Novel Approach for Reduction of Huffman Cost Table in Image Compression”, International Journal of Computer Applications (0975 – 8887) Volume 20– No.6, April 2011.

[3] A Method for the Construction of Minimum-Redundancy Codes David A. Huffman, Associate, IRE. (1952).

[4] Bao Ergude, Li Weisheng, Fan Dongrui, Ma Xiaoyu, " A study and implementation of the Huffman Algorithm based on condensed Huffman table”, Computer Science and Software Engineering, 2008 International Conference on (Volume: 6), ISBN: 978-0-7695-3336-0

[5] Database System Concepts by Silberschatz, Korth, Sudarshan McGraw-Hill Higher Education, ISBN NO.0-07-120413-X.

[4] A Method for the Construction of Minimum-Redundancy Codes DAVID A. HUFFMAN, ASSOCIATE, IRE.

[6] Abraham Sinkov, "Elementary Cryptanalysis: A Mathematical Approach”, Mathematical Association of America, 1966. ISBN 0-88385-622-0.

[7] Ternary Tree & A new Huffman Decoding Technique, IJCSNS International Journal of Computer Science and Network Security, Vol.10 N0.3, March 2010.

[8] A guide to MATLAB by Brian R.Hunt, Ronald L. Lipsman, J.M. Rosenberg Cambridge Univ. Press. ISBN No.0-521- 00859-X.

[9] D. Kahn 1967, “The Code breakers”, The Story of Secret Writing. New York: Macmillan.