Received: December 29, 2018

*International Journal of*
Intelligent Engineering & Systems

http://www.inass.org/

# A High Capacity PDF Text Steganography Technique Based on Hashing Using Quadratic Probing

**Sanjive Tyagi[1]*** **Rakesh Kumar Dwivedi[1]** **Ashendra K. Saxena[1]**

[1]*College of Computing Sciences and Information Technology, Teerthanker Mahaveer University, India*
* Corresponding author's Email: tosanjive@gmail.com

**Abstract:** In this paper, a novel approach of PDF-based text steganography is presented. A covert communication is achieved by embedding secret information at between-characters positions of words within cover PDF text. A stego-encoding technique is being designed to improve the embedding capacity along with sufficiently reduced overheads of stego-cover PDF file. The proposed method constructs multilevel embedding capacities which can be used according to requirements from higher embedding capacity to typical embedding capacity. Encoding and decoding of secret text is being obtained by implementing quadratic probing of hashing technique that builds up a direct encoding table which provides better performance in term of time complexity. Exploratory outcome exhibits that the proposed technique gives productive algorithms in term upgraded security and high payload of concealed data. Experiential verification and examination of the proposed technique with certain prominent text steganographic techniques demonstrate that proposed technique performs better than current methodologies and free of requirements like shading scheme, languages, wordlist, compression and so forth.

**Keywords:** Copyright, Cryptography, Embedding, Steganography, Watermark.

## 1. Introduction

Security is an essential part everywhere in our life to keep the information protected with the objective that any unauthorized couldn't steal them. For this reason, different strategies are available to keep them secured by password security, secret key, finger-printing protection and so forth. However, today in this time of digital communication over the global network, activities like unlawful accessing, tempering, and breaking the copyright act are rising massively. Thus, there is a need to ensure very vital data by utilizing digital security strategies which can be accomplished by covering up confidential digital data inside another digital file or by transforming it into a non-comprehensible frame. In this manner, steganography and cryptography innovations can be assumed a vital part in digital information security framework. Cryptography has its own significance to protect the secret information by making non-understandable, on the other hand, its weakness is encryption information can make the suspicious about its privacy and can be tempered by the third party though preferred standpoint of steganography is, it protects the secret information by embedding it into another digital file thus making the secret information imperceptible so there is less probability of susceptibility. Steganography is a developing research field keeping in mind the goal to provide state-of-the-art development of information security framework [1].

Basic view of PDF layout [2] is being described here such as PDF (Portable Document Format) was created as Acrobat item in 1993 for generating the platform-independent e-document by Adobe (Adobe Systems Incorporated). A PDF file is the arrangement of text, graphics, and binary information. It is an organization of a few indirect objects comprehensively characterized in four segments as (1) header, (2) body, (3) cross-reference section (4) trailer described in Table 1, which comprises of career locations to conceal secret information without influencing the displaying

Table 1. Describing PDF file layout

| | |
|---|---|
| Header | A header recognizes the version of the PDF description to which the file adapts and it comprises of comments. |
| Body | Body section comprises of text objects like Tm, Td, TD operators deals with the text position, operator Tj, operator Tc for character situating, operator Tw for dealing words, operator TJ manage the layout of PDF file which independent of any platform, so on. This section is responsible for the appearance of textual record in PDF file. |
| Cross-reference table | The cross-reference information is used to find any page quickly not subject to the length of a document containing any number of pages. A cross-reference section containing data about the indirect objects in the PDF file. |
| Trailer | A trailer section reference to the area of the cross-reference table and certain unique objects inside the body of the file. |

quality of PDF file. In [3], it is explained that each segment of PDF document is controlled by page formatting depiction script known as PostScript. The method in [3] illustrated Tj administrator strings might be utilized as secret data transporter covertly. There are large integer values being used within Tj operator to describe the style of a text of PDF file. These integer values may be used as the secret information carrier. In this approach controlling the redundancy of integer values of Tj operator is a tedious task at the time of extraction of hidden data. This embedding process may not assure the robustness and high embedding capacity.

PDF text steganography is a valuable approach to cover up secret information. It can be used to convey secret data confidentially across an unreliable network system and to watermark as copyright data into the PDF files. Scheme in [4] described that there are some strategies for text steganography by managing the between-words spaces as one between-words space may imply bit 0 and two between-words spaces may imply 1 and so on., by controlling syntactic rules of language, by controlling semantic guidelines settled between dispatcher and recipient, by making simulating text containing secret words put under specific standards, so on. In most case, it is conceivable in just those word processors where fixed spacing is being used, this is one of the disadvantages of such approaches, whereas portable document file is constructed in a way that white-space characters are utilized to isolate syntactic builds from each other. As indicated by the PDF standard, numerous different characters are dealt with as white spaces. All white-space characters are equal, with the exception of comments, strings, and streams. This property can be utilized to embed secret messages covertly into PDF files.

In [5], the author suggested PDF text steganography by encoding each character of secret text by Huffman coding algorithm thereby null A0s are implanted to conceal secret characters at between-characters positions of words within the cover PDF file. It is noticed during the study this technique embeds the table of frequency also, generated by Huffman coding algorithm in the cover PDF file which increases the size of the cover PDF file and time-consuming. Our paper introduced a stego-encoding technique based on hashing method that may be employed to hide sufficient larger number of secret textual data inside PDF file with better time complexity and fewer overheads than the method in [5].

A semantic text steganography suggested by [6] that employing equivalent word substitution method to hide bit frame of secret text. Here substitution method is changing the actual text which may recognizable as unnatural by the human spectator. In [7], the author presented LZW compression-based text steganography with email-ids and email texts as career medium of secret data. In [8], text steganography presented by combining the move-to-front, burrows-wheeler-transform and LZW compression methods to obtain improved compression of secret text whereas the cover medium is similar to [7]. In [9], an extension of text steganography [8] is presented in order to improve embedding capacity by the application of the Huffman algorithm. In [7-9] schemes cover text chosen is not natural text that utilizing the set of email-ids as a cover medium may create susceptibility which may be noticeable as unnatural by the human observer and can be tempered. An approach of text steganography suggested by [10] utilizing a set of email-ids as career locations of secret text. Furthermore, in the second method of [10] confidential bits are implanted into email body by utilizing the colour shading scheme. These methods use too much email-id as career media also altering the appearance of actual cover contents by colour shading which leads to vulnerability for steganalysis attack. However, our proposed work well against such discrepancies.

Our innovative approach performs better than various existing steganographic schemes in terms that no alteration takes place in the contents of the cover file. It additionally performs well in opposing the steganalysis attack because of invisible control ASCII code A0 are utilized to conceal secret text rather than bits. By introducing hashing technique with quadratic probing the better time complexity is being developed. The arrangement paper is composed as given: In Section 2 related work; Section 3 proposed steganography technique in details with its subsections; section 4 results and discussion; section 5 Comparative Analysis; the conclusion of paper in section 6.

## 2. Related work

The work presented in [11] introduces the text steganography by using the incremental updates feature of PDF document that is generally used to control modification within PDF file, these characteristics of incremental updates is employed to implant confidential data inside PDF file. The incremental update method is appropriate for smaller embedding capacity. An approach in [12] described that OpenPuff scheme implants secret contents within the end-of-line symbols which are referred to as white-space-characters. In this [12], an analytical study was presented that a steganography using popular OpenPuff tool can also be detected which implies that a high embedding capacity scheme is not only the parameter for better performance of steganography technique. Threshold secret sharing scheme is presented in [13] that utilize the cryptographic hash procedure to validate the shared information by an authorized shareholder through implementing the secret sharing method of [14]. This scheme is suitable to decompose and reconstruct the secret data among multiple participants with strong security. The method in [15] introduced a technique by applying a homomorphic encryption method with secret text. This approach has a restriction that cover text should consist of encrypted characters of secret text which have to be embedded whereas our proposed work does not have such restrictions. In [16], the author proposed a steganography scheme in which text of host file and text of secret text is generated by utilizing the process of Markov chain. This dependence of cover contents with secrets text leads susceptibility to steganalysis attack. The scheme presented in [17], introduces the PDF steganography in which secret information is embedded into new entry line of page object Tm for which every time cross-reference and trailer have to be modified otherwise PDF document will not be displayed correctly. Such PDF Steganography is very much time consuming and embedded data may not be extracted properly, however hiding capacity is high. A text steganography method was introduced in [18] that employing transformation of Bengali text for concealing confidential text. They claim this method is safe to oppose steganalysis assaults; however, the transformation of cover contents is always against the imperceptibility.

An approach of text steganography suggested in [19] using missing letter technique, wordlist strategy and by means of a first and last letter of words. In [20], text-based steganography is proposed in which applicant data are substituted by similar outset, including alternative word (synonym), homonym or exceptional character. A Notes-based steganography technique presented in [21] that utilize automatic notetaking strategies by using varieties between input human-notes and produces programmed notes, based on this strategy, secret data are implanted by implementing the substitution method. These substitution methods [19-21] are proposed by changing the features of character, words, and special symbols etc. which make cover contents unnatural that can lead to susceptibility. An approach in [22] demonstrated that short message service provides secured and higher embedding scheme. It is found in study its extraction process is not sufficiently efficient so that it can extract accurate short text message.

In conventional text steganographic methods, some kind of alteration takes place such as substitution by similar outset, wordlist, alternative word, exceptional character, colour shading etc. to the contents of the cover file which may identify as unnatural by a human observer, which leads to susceptibility for steganalysis attack. To overcome this issue invisible control ASCII code A0s are proposed to encode and embed the secret data confidentially within PDF document. Furthermore, between-characters locations of words are used as placeholders for implanting secret data which improve the embedding capacity and imperceptibility. Proposed hashing technique also decreases the overheads of stego-cover text with a sufficiently high payload. The PDF text steganography is being developed as a part of a proposed paper to conceal the secret data inside PDF text file without affecting the visual quality of PDF document.

## 3. Proposed steganography technique

In this section, various methods associated with the proposed steganography technique are discussed.

There are various ASCII codes used to form the PDF text. One of them is non-breaking space i.e. ASCII code A0 is a control code. ASCII code A0 have a characteristic, when embedded with any

Table 2. Hash-Value-Array [126] in Table 3 is divided into three sections based on the given range of ASCII code

| Hash-table Number | The range of ASCII code | Size of table (n) taken nearest prime number | Binary Code to represent table number |
|---|---|---|---|
| HTable1 | 48 to 90 | 43 | 001 |
| HTable2 | 91 to 126 | 36 ≈37 | 010 |
| HTable3 | 10, 32 to 47 | 17 | 011 |

ASCII character then it creates invisible white space and it occupies the width in the text equivalent to single space but when its width is made to zero called zero-width A0, it is possible because PDF format permits to control the width assessment of every character used in PDF text. In proposed approach ASCII code A0 does not occupy the width of one character thereafter it became space-less without width but zero-width A0 is there, hence it can be embedded between-characters where it is invisible without occupying the visible space when it is seen from the normal view of PDF file. This is just like sandwiching A0 without occupying any width between characters of a cover text of PDF file. We have proposed a new zero-width coding technique based on hashing using quadratic probing that may be employed to hide a larger number of secret textual data in PDF file with better time complexity.

## 3.1 Method of encoding characters to hash-values

This section briefly describes the hashing scheme used to encode characters of secret data.

In the proposed method, there are only 96 printable characters generally used to make the secret text. ASCII codes of printable characters lie between 10, 32 to 126 including LF (line feed) as controlling printable character. These ASCII codes of printable characters are divided into three sub-tables (shown in Table 2) in order to get smaller hash value corresponding to each ASCII character which will be used as the frequency of zero-width's A0s to embed the encoded characters of secret text at between-characters positions of words within the cover PDF file. Therefore sub tables of ASCII characters are designed as Table 1.

Hashing technique is being designed to encode the ASCII code of each printable characters of secret text, where hash function using Quadratic Probing is taken [23] in a given form:

$h\ (key, j) = (h'(key) + x_1.j + x_2.j^2) \bmod n$, here h denotes hash function for a value of key of an auxiliary type, $x_1$ !=0 and $x_2$ !=0, $x_1$ and $x_2$ are constants of auxiliary type and j= 0,1, 2. 3, … , n-1.

### 3.1.1 Encoding based on hash function using quadratic probing

We have hash function using Quadratic Probing

$$h\ (key, j)=[(key \bmod n) + x_1.j + x_2.\ j^2) \bmod n, \qquad (1)$$

where key = ASCII code within the range of ASCII code shown in Table 2, n=43 for Hash-table1, n=37 for Hash-table2, n=17 for Hash-table3, $x_1$=1 and $x_2$=3 are taken arbitrarily. Using hash function Eq. (1), one hash value equals to 0 is obtained for one ASCII code in each sub table. Here the hash value of ASCII character equals to 0 cannot represent the ASCII code of any printable character so replace hash value 0 with its n value.

### 3.1.2 Compute Hash-Value

In the proposed method, secret data is made up 96 printable characters, which ASCII code is given as 10, 32 to 126 and 1-D array named Hash-Value-Array [size] is created where size is 126 shown in Table 3 referred as direct encoding table, here array indexes are the ASCII code of 96 printable characters additionally corresponding values of Hash-Value-Array [ASCII-Code] are hash values computed using the following algorithm Compute Hash-Value.

**Input**: ASCII code of characters
**Output**: Hash-value is a number
**Algorithm Compute Hash-Value**

```
1. x1=1, x2=3
2. if ASCII-code >=48 and ASCII-code < =90 then
3. Compute hash-value = [ (ASCII-code mod n) + x1.j + x2.j^2 ] mod n, using quadratic probing for n=43, j= 1… n
4. if hash-value !=0 then Hash-Value-Array[ASCII-code]=hash-value else if hash-value = 0 then Hash-Value-Array[ASCII-code]=n
   endif // endif of line 4
   endif // endif of line 4
   endif // endif of line 2
5. if ASCII-code >=91 and ASCII-code< =126 then
6. Compute hash-value= [(ASCII-code mod n) + x1.j + x2.j^2 ] mod n using quadratic probing for n=36, j= 1… n
7. if hash-value!=0 then Hash-Value-Array[ASCII-code]=hash-value else if hash-value=0 then Hash-Value-Array[ASCII-code]=n
   endif // endif of line 7
   endif // endif of line 7
```

```
    endif // endif of line 5
8.  if ASCII-code >=10 and ASCII-code < =47 then
9.  Compute hash-value= [ (ASCII-code mod n) + $x_1.j$ +$x_2.j^2$ ] mod n using quadratic probing for n=17, j= 1… n
10. if hash-value !=0 then Hash-Value-Array[ASCII-code]=hash-value else if hash-value=0 then Hash-Value-Array[ASCII-code]=n
    endif // endif of line 10
    endif // endif of line 10
    endif //endif of line 8
11. end
```

## 3.2 Implementation of proposed steganography technique

This section briefly describes details of proposed embedding and extracting process.

### 3.2.1 Steps for embedding secret data

In this subsection, steps for embedding secret text into cover PDF text file are discussed using proposed zero-width Coding Technique.

Step 1: In order to obtain a balanced size of stego-cover PDF file we choose to hide one character within one between-characters location per word and referenced hash table number is embedded at followed between-words space. For convenient, it is assumed that number of hidden characters minus one equal to the number of words in the cover PDF file, where the last word is not used as place-holder of secret character which indicates the end of file of stego-cover PDF file.

Step 2: Count number of characters in secret text file and transform each character of secret text into its equivalent ASCII code and store them into a matrix named SCASCII[nRow][nCol] as follows SCASCII[nRow][nCol] = { encodeToASCII(Secret text file) }, where nCol is number of columns (64), nRow (number of rows) is total number of secret characters/64.

Step 3: Each cell of matrix SCASCII[nRow][nCol] have to be embedded into cover PDF file named CoverPdf by implementing the Algorithm for Embedding Secret Data, by applying zero-width coding technique. (FrequencyA0s is a variable to store a hash value corresponding ASCII code)

**Algorithm for Embedding Secret Data-**

**Input:** Cover PDF file named CoverPdf and secret data SCASCII[nRow][nCol] in matrix form, where nCol is number of columns (64), nRow (no. of rows) is total number of secret characters/64.

**Output:** CoverPdfStego as stego cover file

**Algorithm Embedding Secret Data**

```
1.  Open CoverPdf PDF file on read and write mode to scan and edit CoverPdf PDF file
2.  CountSecretCharacters ← Count number of characters in secret text file
3.  CountCoverWords ← Count number of words in CoverPdf file
4.  CountCoverWords ← CountCoverWords -1 // last word CoverPdf file will not be used
5.  if CountSecretCharacters > CountCoverWords then
        Display "Secret text file is larger in size"
        exit
    endif
6.  nCol ← 64
7.  nRow ← CountSecretCharacters/64
8.  Transform each character of secret text into ASCII code and store them into matrix SCASCII[nRow][nCol]
9.  Locate between-characters location of word to $l_c$ in CoverPdf file // consider first between-characters location of each word is being used to embed encoded secret character
10. CtrRow ← 1 //indicating row no.
11. while (CtrRow < = nRow)
12. CtrCol ← 1 // indicating column no.
13.   while (CtrCol < = nCol)
14.   FrequencyA0s ← Hash-Value-Array [ SCASCII [ CtrRow ] [ CtrCol ] ]
15.   HtableNo ← Call TableNumber ( SCASCII [CtrRow][CtrCol] )
16.   Embed no. of zero-width A0s as FrequencyA0s no. (obtained from step 14) at location $l_c$ of CoverPdf file
17.   Locate between-words space in CoverPdf file then embed no. of zero-width A0s as HtableNo no. at located between-words space // embed 1 A0 for Htable no=1, 2 A0s for Htable no=2, 3 A0s for Htable no=3
18.   CtrCol ← CtrCol + 1
19.   Get between-characters location to $l_c$ in next word of CoverPdf file
      endwhile
20. CtrRow ← CtrRow + 1
21. endwhile
22. close CoverPdf file
23. rename CoverPdf file as CoverPdfStego file
24. End
```

**Algorithm to find hash table no-**

This algorithm is used to find sub-table number to which secret character's ASCII code belongs to

**Algorithm Find TableNumber ( ASCII-code)**

```
1.  If ASCII-code >=48 and ASCII-code < =90 then
```

```
    HTableNo=1
    endif
2.  If ASCII-code >=91 and ASCII-code< =126 then
    HTableNo=2
    endif
3.  If ASCII-code >=10 and ASCII-code < =47 then
    HTableNo=3
4.  endif
5.  return HTableNo
6.  End
```

**Running time-** Let us assume secret text file consists of n number of secret characters thus time complexity of embedding algorithm is dependent on number of characters of secret text file and procedure Find-TableNumber takes O(1) time and step 14 (obtaining FrequencyA0s i.e. hash value) in above algorithm is using direct address table takes O(1) time to encode targeted embedding character, hence time complexity of algorithm = *O(n) + O(1) + O(1) = O(n).*

### 3.2.2 Steps for extracting of secret data

This subsection briefly describes steps for extracting the secret text from stego-cover PDF text file using proposed zero-width Coding Technique.

Step 1: In order to extract ASCII codes of secret characters, create a matrix ESCASCII[nRow][nCol], where nRow and nCol are obtained from valid stego-cover PDF file. Then obtain the indexes by searching the hash values within Hash-Value-Array [126] (shown in Table 3), where hash values are equals to a number of zero-width's A0s extracted from between-characters positions of each word inside stego-cover PDF file. These obtained indexes are representing ASCII code of secret characters which will be stored into a created matrix by implementing the Algorithm for Extracting Secret Data in Step 2.

Step 2: Extract and decode the embedded secret characters by extracting the zero-width's A0s by applying zero-width decoding technique as the given algorithm.

**Input:** Stego cover PDF file named CoverPdfStego.
**Output:** Matrix ESCASCII[nRow][nCol] of ASCII code of extracted secret characters, where nCol is number of columns (64), nRow (number of rows) is (total number of words in CoverPdfStego -1)/64.

**Algorithm for Extracting Secret Data**

```
1.  Open CoverPdfStego PDF file on read mode to
    scan CoverPdfStego file
2.  nw ← Count number words of CoverPdfStego
    file -1
3.  nCol ← 64
4.  nRow ← nw/nCol
5.  Declare matrix ESCASCII[nRow][nCol]
6.  Locate word in CoverPdfStego file
7.  keyWord ← Get located word stream
8.  Locate between-words space
9.  keySpace ← Get located between-words space
    stream
10. CtrRow ← 1 //indicating row no.
11. CtrCol ← 1 //indicating column no.
12. while not eof (CoverPdfStego)
13.   EHashValue ← Find and count number of
      zero-width's A0s between-characters of
      keyWord
14.   ETableNo ← Count number of zero-width's
      A0s in keySpace
15.   EASCII-Code ← Call SearchASCIIValue
      (EHashValue, ETableNo)
16.   ESCASCII[CtrRow][CtrCol] ← EASCII-
      Code
17.   Locate next word in CoverPdfStego file
18.   keyWord ← Get located word stream
19.   Locate next between-words space
20.   keySpace ← Get located between-words space
      stream
21.       If eof (CoverPdfStego) then
22.       Exit from while loop
          endif
23.       if CtrCol = 64 then
24.       CtrCol ← 0
25.       CtrRow ← CtrRow +1
26.       endif
27.   CtrCol ← CtrCol +1
28. endwhile
29. end
```

**Output:** Extracted secret text is obtained.
**Running Time:** Let us assume cover file consists of n number of words then its running time is based on the number of words within cover file plus running time of procedure SearchASCIIValue which is a constant, hence time complexity of this algorithm is *O(n)+ O(1) =O(n).*

**Procedure SearchASCIIValue (EHashValue, ETableNo)**

This procedure (algorithm) searches EHashValue in the array Hash-Value-Array [126] shown in Table 3 containing three sections as HTable1, HTable2, and HTable3, where section of table is decided on the basis of ETableNo. If EHashValue is found in an array as given algorithm then it returns the index of found EHashValue, which is actually the ASCII code of the desired character then obtained ASCII code can be decoded to the readable secret character.

Table 3. Direct encoding table- obtained from hashing technique using quadratic probing: Secret characters are encoded by obtaining their hash values using Hash-Value-Array [ASCII-code]

| **HTable No= 3** | | **HTable No = 1** | | | | **Htable No = 2** | | | |
|---|---|---|---|---|---|---|---|---|---|
| Hash-Value-Array from index 1 to 47 consist of n=17 values | | Hash-Value-Array from index 48 to 90 consist of n=43 values | | | | Hash-Value-Array from index 91 to 126 consist of n=36 values | | | |
| Array Index | Hash-Value | Array Index | Hash-Value | Array Index | Hash-Value | Array Index | Hash-Value | Array Index | Hash-Value |
| 10 | 10 | 48 | 5 | 70 | 27 | 91 | 19 | 113 | 5 |
| 32 | 15 | 49 | 6 | 71 | 28 | 92 | 20 | 114 | 6 |
| 33 | 16 | 50 | 7 | 72 | 29 | 93 | 21 | 115 | 7 |
| 34 | 17 | 51 | 8 | 73 | 30 | 94 | 22 | 116 | 8 |
| 35 | 1 | 52 | 9 | 74 | 31 | 95 | 23 | 117 | 9 |
| 36 | 2 | 53 | 10 | 75 | 32 | 96 | 24 | 118 | 10 |
| 37 | 3 | 54 | 11 | 76 | 33 | 97 | 25 | 119 | 11 |
| 38 | 4 | 55 | 12 | 77 | 34 | 98 | 26 | 120 | 12 |
| 39 | 5 | 56 | 13 | 78 | 35 | 99 | 27 | 121 | 13 |
| 40 | 6 | 57 | 14 | 79 | 36 | 100 | 28 | 122 | 14 |
| 41 | 7 | 58 | 15 | 80 | 37 | 101 | 29 | 123 | 15 |
| 42 | 8 | 59 | 16 | 81 | 38 | 102 | 30 | 124 | 16 |
| 43 | 9 | 60 | 17 | 82 | 39 | 103 | 31 | 125 | 17 |
| 44 | 14 | 61 | 18 | 83 | 40 | 104 | 32 | 126 | 18 |
| 45 | 11 | 62 | 19 | 84 | 41 | 105 | 33 | | |
| 46 | 12 | 63 | 20 | 85 | 42 | 106 | 34 | | |
| 47 | 13 | 64 | 21 | 86 | 43 | 107 | 35 | | |
| | | 65 | 22 | 87 | 1 | 108 | 36 | | |
| | | 66 | 23 | 88 | 2 | 109 | 1 | | |
| | | 67 | 24 | 89 | 3 | 110 | 2 | | |
| | | 68 | 25 | 90 | 4 | 111 | 3 | | |
| | | 69 | 26 | | | 112 | 4 | | |
| Average-Hash-Value=9 | | Average-Hash-Value=22 | | | | Average-Hash-Value=18 | | | |
| Average-Hash-Value of HTable No-1, HTable No-2 and HTable No-3 = 16.33 | | | | | | | | | |

**Algorithm SearchASCIIValue (EHashValue, ETableNo)**

```
1. if ETableNo = 1 then search EHashValue in Hash-
   Value-Array between array index 48 and 90 then
   if EHashValue found in an array
2.     then return index of found EHashValue
       endif
   endif
3. if ETableNo = 2 then search EHashValue in Hash-
   Value-Array between array index 91 and 126 then
   if EHashValue found in an array
4.     then return index of found EHashValue
       endif
   endif
5. if ETableNo = 3 then search EHashValue in Hash-
   Value-Array between array index 10 and 47 then
   if EHashValue found in an array
6.     then return index of found EHashValue
       endif
   endif
7. end
```

Step 3: Transform each cell of matrix ESCASCII[nRow][nCol] into equivalent character corresponding to its ASCII code, hence obtain the extracted hidden secret information from CoverPdfStego cover PDF file.

Secret Text
= {ASCII-decode (ESCASCII [nRow] [nCol] )}

**Output:** Extracted secret data is obtained.

## 4. Results and discussion

### 4.1 Performance analysis

Here we are analyzing the performance of our proposed work by computing its time complexity and comparing it with various existing schemes.

In the proposed scheme any printable character is encoded by obtaining the hash value in one step corresponding to its ASCII code by using direct encoding table shown in Table 3, hence this searching operation takes O(1) constant time so if there are n number of character to get the hash values then time complexity of proposed scheme would be O(n) whereas time complexity of Huffman algorithm is O(n log n) for encoding the characters used by [5]. At the time of extraction, the time complexity is also O(n) since n number secret characters have to be decoded from n number of words including running time of procedure SearchASCIIValue which is also constant. Therefore the proposed encoding approach is better in term of time complexity. In the proposed approach there is no need to embed the hashing table with secret data because it can be regenerated under predefined rules between sender and receiver. However, in [5] it is required to embed the Huffman coding table as part of hidden data in order to recover secret data towards recipients' side.

## 4.2 Experimental result

Here, we are exhibiting simulation outcomes by inspecting and validating the proposed approach with sample secret text and sample cover PDF file. Let us assume a sample cover PDF text file on an average contains 51 lines per page, 10 words per line, 510 words per page, and 6.5 characters per word with all type of average margins 1.2 inch, font size 11pt, single line spacing where size of one character is taken one byte and measurement is considered of only sample text.

Let on an average N is 510 be the number of words per page in a cover file C. As it is assumed on an average each word consist of Nwc is 6.5 number of characters then total number of characters in our sample page denoted by Nctot including words and between-words spaces are taken as *Nctot = N x Nwc + (N-1) =510 x 6.5+509= 3824.00.* Here between-words spaces are not focused locations to hide the secret text. Destined between-characters positions per word denoted by *Nbc = (Nwc-1) = 5.5* then targeted career locations denoted by Nc per sample page can be obtained by *Nc=(N-1) x Nbc = 509 x 5.5 = 2799.50* as the last word of sample page or file is not used to hide secret text which indicate end of stego-word in file. However between-words spaces are used to hide additional secret data that is referenced to a hash-table number. In the proposed approach encoding of secret characters is achieved by direct encoding hash techniques by using quadratic probing scheme. During the study, we found that if we use all between-characters locations then the size of cover text increases with high proportion which is not appropriate for the sake of security which may create a suspicious for steganalyst. Analysis has been done as shown in Table 4 and Fig. 1. Here it is found varying embedding capacities can be achieved using proposed approach. It is evaluated that normal (typical) embedding capacity 1.66% is obtained by choosing one between-characters position per 8th word, medium embedding capacity 13.31% is obtained by choosing one between-characters position per word and higher embedding capacity 73.21% is obtained by choosing each position within each word as shown in Table 4. Embedding gains are investigated in Fig. 1 to determine the rate of increase in the size of stego-text with proportion to selecting the between-characters positions within words of a cover PDF document. It is found in experiential analysis that two preferable cases, first is, one between-characters position per word where 509 bytes of secret text is embedded into cover text of size 3824.00 bytes then embedding capacity is evaluated to 13.31% and second is, fourth between-characters position per word where 699.88 bytes of secret text is embedded into cover text of size 3824.00 bytes then embedding capacity is evaluated to 18.30%. These are two cases where the size of stego-cover text file is not growing unexpectedly. It is clear from Fig. 1 that case 1 is best suited as it lies in between of all other cases.

Table 4. Experimental outcomes of the proposed scheme using sample text and sample cover PDF file

| | | **Unit of size measurement in bytes** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **SN** | **Chosen between-characters position of cover text to hide secret data** | **Size of cover text** | **Targeted positions available in cover text** | **Payload of Secret Text** | **Size of Stego text** | **Growth in size of Cover text** | **Growth in size of Cover text %** | **Embedding capacity %** |
| 1 | Each position in word | 3824.00 | 2799.50 | 2799.50 | 49541.84 | 45717.84 | 1195.55 | 73.21 |
| 2 | Each 2nd position in word | 3824.00 | 1399.75 | 1399.75 | 26683.92 | 22859.92 | 597.80 | 36.60 |
| 3 | Each 4th position of word | 3824.00 | 699.88 | 699.88 | 15254.96 | 11430.96 | 298.93 | 18.30 |
| 4 | One position per word | 3824.00 | 509.00 | 509.00 | 12137.97 | 8313.97 | 217.42 | 13.31 |
| 5 | One position per 2nd word | 3824.00 | 254.50 | 254.50 | 7981.99 | 4157.99 | 108.73 | 6.66 |
| 6 | One position per 4th word | 3824.00 | 127.25 | 127.25 | 5903.99 | 2079.99 | 54.39 | 3.33 |
| 7 | One position per 8th word | 3824.00 | 63.63 | 63.63 | 4865.00 | 1041.00 | 27.22 | 1.66 |
| Average number of zero width A0 needed to represent any character is computed 16.33 | | | | | | | | |

Table 5. Comparison of embedding (hiding) capacity among existing and proposed technique

| **Methods Used** | **Capacity (%)** | **Explanation** |
|---|---|---|
| [19] PA(Paragraph Approach ) | 2.151 | Evaluated by employing the dataset sample-2 in Table 6 |
| [7] | 6.925 | Evaluated by employing the dataset sample-2 of referred article in Table 6 |
| [8] | 7.030 | Evaluated by employing the dataset sample-2 of referred article in Table 6 |
| [9] | 7.210 | Evaluated by employing the dataset sample-2 of referred article in Table 6 |
| [19] MLA(Missing Letter Approach) | 8.672 | Evaluated by employing the dataset sample-2 in Table 6 |
| [6] | 9.100 | Reported by basing on the sample in the referred articles |
| [10] | 10.891 | Evaluated by employing the dataset sample-1 of referred article in Table 6 |
| Proposed approach | 13.310 | Evaluated by employing the dataset sample-3 in Table 6 |

Table 6. Training datasets used (without quotes) to evaluate embedding capacity for comparative analysis

| | **Secret data** | | **Cover Media** | | |
|---|---|---|---|---|---|
| **Dataset** | Secret text | No. of Characters | Cover text | Text type | No. of Characters |
| **Sample-1** | "underlying ……. Mechanisms" | 33 | "you cant …… yourself" and 5 email ids as "qheet@btinternet.com, vsert@aol.com,......" | plain text | 51 |
| **Sample-2** | "behind using…………… intended recipient" | 198 | "in the research area ……………least 16 bits" | plain text | 847 |
| **Sample-3** | "Information can be hidden……… dependent" | 509 | "Steganography is the technology ……… immensely" | PDF text | 3824 |

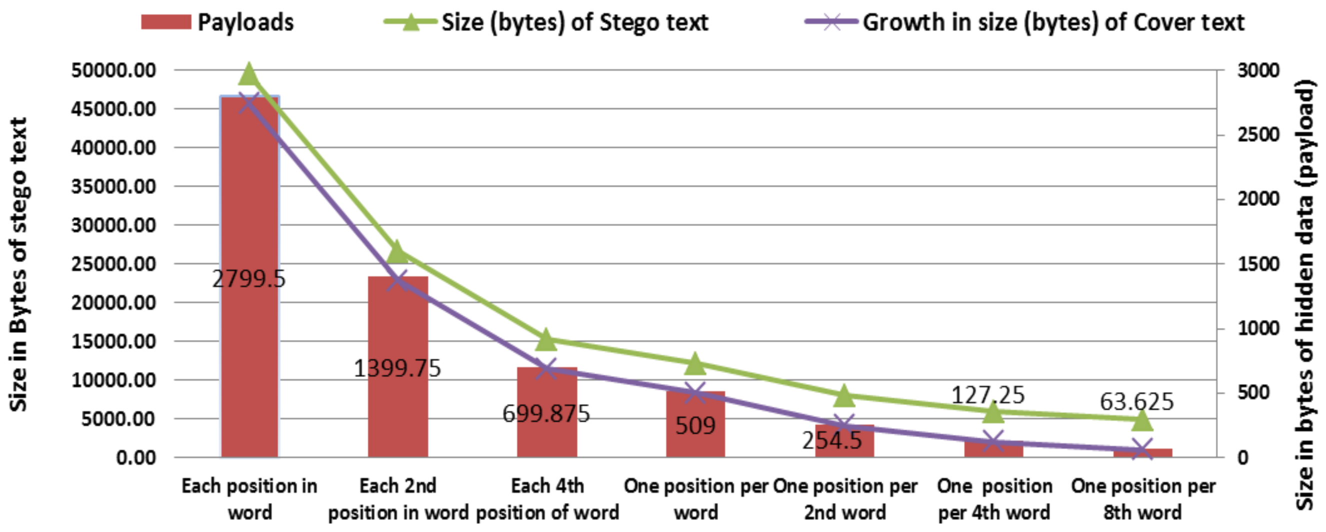

Figure. 1 Observing payloads (size in bytes of hidden data) based on selecting between-characters positions within words of cover text (PDF file), locations used within words of cover text to hide secrete characters

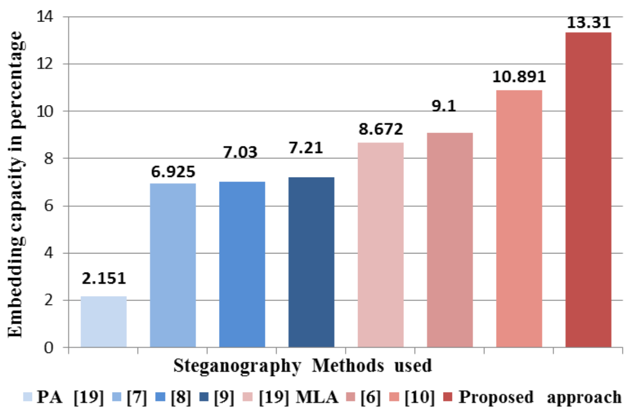

Figure. 2 Comparison of embedding capacities among existing techniques and proposed technique

## 5. Comparative analysis

Embedding capacities of various existing schemes and proposed technique are evaluated in Table 5 and training datasets used are shown in Table 6. Experiential analysis and verification of the proposed system with certain prominent text steganographic procedures exhibited in Table 5 and Fig. 2 respectively. Here outcomes developed that our proposed scheme giving significantly higher embedding capacity. In the event that we compare our proposed approach with the method that utilizing encoding of secret data by colouring pattern within cover message then it is discovered, excessively shading combination utilized as a part of content can create suspicious which is not suggestible however our projected approach does not make any such suspicious. If we compare proposed approach with steganography based on languages then it is found phonetic steganography frameworks begin with cover content, thereafter, this cover message is being altered to conceal secret data this alteration offers to ascend to specific vulnerabilities. One kind of weakness is that the altered content is identifiable as unnatural by the human observer, whereas our presented scheme is free from requirements like shading scheme, languages, wordlist, alteration, compression etc. that exhibit predominance of our proposed approach.

## 6. Conclusion

This paper introduced an innovative implementation process with effective embedding and extracting algorithms which transforming secret text into a highly protected encoded frame with high payload steganography. A novel stego-encoding technique is being introduced that produces less number embedding bits, that lessens the gain in size of stego-cover file with high embedding capacity that likewise reduces the overheads of stego-cover PDF file. Here interesting feature of this scheme is that embedding capacity can be managed by manipulating a number of targeted between-characters positions. In light of our outcomes, we trust that adequate high embedding capacity 13.31% might be acquired by choosing one between-characters position per word. Execution assessment and result examination exhibits that presented steganographic technique is secure besides the implanting capacity is adequately improved and this innovative approach is practically feasible. Further research on this paper might be extended to exceptionally secured image steganography incorporated with PDF-based steganography.

## References

[1] S. Tyagi, A. K. Saxena, and S. Garg, “Secured High Capacity Steganography using Distribution Technique with Validity and Reliability”, In: *Proc. of the International Conf. on System Modeling & Advancement in Research Trends*, pp.109 –114, 2016.

[2] Adobe System Incorporated, PDF Reference Sixth edition, Version 1.7, http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/pdf_reference_1-7.pdf, November 2006.

[3] S. Zhong, X. Cheng, and T. Chen, “Data Hiding in a Kind of PDF Texts for Secret Communication”, *International Journal of Network Security*, Vol. 4, No. 1, pp.17–26, 2007.

[4] D. Salomon, “Data Hiding in Text”, *eBook of Data Privacy and Security*, Chapter-10, pp-247-267, 2003.

[5] I. S. Lee and W. H. Tsai, “A New Approach to Covert Communication Via PDF Files”, *Signal Processing*, Vol. 90, No. 2, pp. 557-565, 2010.

[6] S. Mahato, D. A. Khan, and D. K. Yadav, “A Modified Approach to Data Hiding in Microsoft Word Documents by Change-Tracking Technique”, *Journal of King Saud University – Computer and Information Sciences*, In Press, 2017.

[7] E. Satir and H. Isik, “A Compression-Based Text Steganography Method”, *Journal of Systems and Software*, Vol. 85, No. 10, pp. 2385-2394, 2012.

[8] R. Kumar, S. Chand, and S. Singh, “An Email Based High Capacity Text Steganography Scheme using Combinatorial Compression”, In: *Proc. of the International Conf. on Confluence-*

*The Next Generation Information Technology Summit*, pp. 336–339, 2014.

[9] R. Kumar, S. Chand, and S. Singh, "A High Capacity Email Based Text Steganography Scheme using Huffman Compression", In: *Proc. of the International Conf. on Signal Processing and Integrated Networks*, pp. 53-56, 2016.

[10] A. Malik, G. Sikka, and H. K. Verma, "A High Capacity Text Steganography Scheme Based on LZW Compression and Color Coding", *Engineering Science and Technology, an International Journal*, Vol. 20, No. 1, pp. 72-79, 2017.

[11] H. Liu, L. Li, J. Li, and J. Huang, "Three Novel Algorithms for Hiding Data in PDF Files Based on Incremental Updates", *eBook* of *Digital Forensics and Watermarking,* Vol. 7128, pp.167–180, 2012.

[12] T. Sloan and J. H. Castro, "Dismantling OpenPuff PDF Steganography", *Digital Investigation*, Vol. 25, pp. 90-96, 2018.

[13] K. M. Faraoun, "A Novel Fast and Provably Secure (T, N)-Threshold Secret Sharing Construction for Digital Images", *Journal of Information Security and Applications*", Vol. 19, No. 6, pp. 331-340, 2014.

[14] A. Shamir, "How to Share a Secret", *Communication of the ACM*, Massachusetts Institute of Technology, Vol. 22, pp. 612-613, 1979.

[15] N. Naqvi, A. T. Abbasi, R. Hussain, M. A. Khan, and B. Ahmad, "Multilayer Partially Homomorphic Encryption Text Steganography (MLPHE-TS): A Zero Steganography Approach", *Wireless Personal Communications*, pp. 1-23, 2018.

[16] A. N. Shniperov and K. A. Nikitina, "A Text Steganography Method Based on Markov Chains", *Automatic Control and Computer Sciences,* Vol. 50, No. 8, pp. 802–808, 2016.

[17] Y. C. Lai and W. H. Tsai, "Covert Communication via PDF Files by New Data Hiding Techniques", In: *Proc. of Conf. on Computer Vision, Graphics and Image Processing*, 2009.

[18] M. Khairullah, "A Novel Steganography Method using Transliteration of Bengali Text", *Journal of King Saud University – Computer and Information Sciences,* In Press, 2018.

[19] M. Agarwal, "Text Steganographic Approaches: A Comparison", *International Journal of Network Security & Its Applications*, Vol. 5, No. 1, pp. 91-106, 2013.

[20] L. Gongshen, D. Xiaoyun, S. Bo, and M. Kui, "A Text Information Hiding Algorithm Based on Alternatives", *Journal of Software*, Vol. 8, No. 8, pp. 2072-2079, 2013.

[21] A. Desoky, "Notestega: Notes-Based Steganography Methodology", *Information Security Journal: A Global Perspective*, Vol. 18, No. 4, pp. 178-193, 2009.

[22] W. Ren, Y. Liu, and J. Zhao, "Provably Secure Information Hiding via Short Text in Social Networking Tools", *Tsinghua Science and Technology*, Vol. 17, No. 3, pp. 225-231, 2012.

[23] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms", *MIT press, Publisher: PHI Learning Pvt. Ltd*, New Delhi, India, 2009.