RESEARCH ARTICLE OPEN ACCESS

Proposed Arabic Text Steganography Method Based on New Coding Technique

Assist. prof. Dr. Suhad M. Kadhem*, Dhurgham Wameedh**
**Computer Science Department/ University of Technology/ Baghdad/ Iraq*
*** Computer Science Department/ University of Technology/ Baghdad/ Iraq*

ABSTRACT: Steganography is one of the important fields of information security that depend on hiding secret information in a cover media (video, image, audio, text) such that un authorized person fails to realize its existence. One of the lossless data compression techniques which are used for a given file that contains many redundant data is run length encoding (RLE). Sometimes the RLE output will be expanded rather than compressed, and this is the main problem of RLE. In this paper we will use a new coding method such that its output will be contains sequence of ones with few zeros, so modified RLE that we proposed in this paper will be suitable for compression, finally we employ the modified RLE output for stenography purpose that based on Unicode and non-printed characters to hide the secret information in an Arabic text.
Keywords: Security, Steganography, RLE, Coding, Compression.

I. INTRODUCTION

Steganography is the art of concealing private or sensitive information within information that appears to be not raising suspicion. It always gets confused with cryptography because they are similar in functionality, steganography and cryptography used to protect data. Steganography actually involve hiding information but this information not appears to the reader [1]. The objective of steganography is to conceal, deliver messages, taking into consideration the exchange of information [2]. Example of steganography is invisible ink that used for readable message contact between sender and receiver. Any attack can read the message without any clue about the concealing data, but the authorizes persons can read hidden information based on inductions features [3].

In this paper we will use proposed a steganography method that based on modified run length encoding (RLE), Unicode and non-printed characters is also used to embed the secret information because most of these characters do not appear on the screen when written.

This paper is organized as follow: section 2 presents a brief explanation about text steganography, section 3 presents an explanation about RLE, section 4 presents an explanation about non printed characters, section 5 presents Unicode, section 6 presents Related Work, Section 7 presents the proposed steganography method with its algorithms, section 8 describe the proposed method iplementation, section 9 presents the performance analysis and finally section 10 state the main conclusions.

II. TEXT STEGANOGRAPHY

Steganography consist of four classes: audio, video, image and text steganography based on cover media that used to conceal secret information. Text steganography can consist of anything from edit the formatting of an existing text, like replacing word within text, to generating random character sequences or using context-free grammars to generate readable texts [4]. Figure (1) illustrates text steganography idea. Firstly, a secret message will be hiding in a cover-text by applying an embedding algorithm to produce a stego-text. After that the stegotext will transfer by communication channel [5].

There are several types of text steganography: structural, random, statistical generation and linguistic. Structural text steganography contains replace the physical structure of the text, for example by insert whitespace or increasing the line spacing. Random and statistical generation contains generating the covertext either randomly, or according to some algorithms. Linguistic contains process the syntactic or semantic specification of the existing text [6].

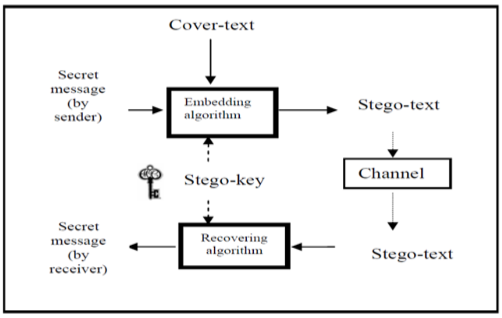

Figure (1): The Mechanism of Text Steganography

III. RUN LENGTH ENCODING (RLE)

It is a compression technique which is used for a given file that contains many redundant data. The input file or message is called run which is encoded into two bytes, the first byte contains the number of times for a given character appears in the run, and the second byte represents the value of the character [7]. If we have a sequence of un identical data then the drawback of this method will be appears, because the data will be expanded rather than compressed, for example if we have a stream of data like 1010 the output of RLE will be **11101110,** so it is expanded rather than compressed, but if we have the a sequence of data like **1111111111111110**, then the output will be **16110**, so the data will be compressed **[**8]. In our proposed coding method we prepare the data in order to be a sequence of identical data to benefit from RLE compression method.

IV. NON PRINTED CHARACTERS

We have several characters which are not normally displayed on the screen. For example, there is a special character to indicate the end of a line or the end of a paragraph, and so on [9]. Table (1) illustrates these characters.

Table (1): Non Printing Characters

Non printing characters					
Des	**Hex**	**Character(Code)**	**Dec**	**Hex**	**Character(Code)**
0	0	NULL	16	10	DATA LINKESCAPE (DLE)
1	1	START OF HEADING (SOH)	17	11	DEVICE CONTROL 1 (DC1)
2	2	START OF TEXT (STX)	18	12	DEVICE CONTROL 2 (DC2)
3	3	END OF TEXT (ETX)	19	13	DEVICE CONTROL 3 (DC3)
4	4	END OF TRANSMISSION (EOT)	20	14	DEVICE CONTROL 4 (DC4)
5	5	END OF QUERY (ENQ)	21	15	NEGATIVE ACKNOWLEDGEMENT (NAK)
6	6	ACKNOWLEDGE (ACK)	22	16	SYNCHRONIZE (SYN)
7	7	BEEP (BEL)	23	17	END OF TRANSMISSION BLOCK (ETB)
8	8	BACKSPACE (BS)	24	18	CANCEL (CAN)
9	9	HORIZONTAL TAB (HT)	25	19	END OF MEDIUM (EM)
10	A	LINE FEED (LF)	26	1A	SUBSTITUTE (SUB)
11	B	VERTICAL TAB (VT)	27	1B	ESCAPE (ESC)
12	C	FF (FORM FEED)	28	1C	FILE SEPARATOR (FS) RIGHT ARROW
13	D	CR (CARRIAGE RETURN)	29	1D	GROUP SEPARATOR (GS) LEFT ARROW
14	E	SO (SHIFT OUT)	30	1E	RECORD SEPARATOR (RS) UP ARROW
15	F	SI (SHIFT IN)	31	1F	UNIT SEPARATOR (US) DOWN ARROW

V. UNICODE SYSTEM STANDARD

Unicode is a standard to encode all of the word's languages correctly on computers .It is an international standard. Its objective is to overcome ambiguities that traditionally arise when displaying complex scripts like Japanese, Arabian or Chinese on computer systems. Traditional character sets (like the American National Standards Institute ANSI alphabet) are depending on (8 bit) letters named a byte. A single byte can represent up to (256) different values and thus letters. This is well enough to represent western scripts like that being used in English, French or German language. However, if it gets to more complex languages like Japanese or Korean, (256) different letters are simply not enough [10].

In our paper we use two Unicode with ASCII 157 and 158, the function of Unicode with 158 ASCII is to convert the isolated Arabic character to connected Arabic character, so in our proposed steganography method we embed it with the connected character to not make any change to the Arabic cover text but it give an indication to the receiver, the same thing happened with the Unicode 157 ASCII that convert the connected character to isolated character, so we embed it with the isolated character.

VI. RELATED WORKS

In this section we will investigate some of steganograpgy methods that related to our work:

- **A Novel Arabic Text Steganography Method Using Letter Points and Extensions, 2007 [11].**

Adenan and manal present a new steganography approach suitable for Arabic texts. It can be classified under steganography feature coding methods. The approach hides secret information bits within the letters benefiting from their inherited points. To note the specific letters holding secret bits, the scheme considers the two features, the existence of the points in the letters and the redundant Arabic extension character. We use the pointed letters with extension to hold the secret bit ‘one’ and the un-pointed letters with extension to hold ‘zero’. This steganography technique is found attractive to other languages having similar texts to Arabic such as Persian and Urdu.

- **A New Text Steganography Method By Using Non-Printing Unicode Characters", 2010[12].**

This is an approach for text steganography by using Unicode standard characters, (which have the non-printing properties) to coding the letters of English language and embedding the secret message letter by letter into the cover-text. This approach has high hiding capacity , it can embed (K+1) letters in a text with K characters and it does not make any apparent changes in the original text. So it satisfies perceptual transparency.

- **Information Hiding in Arabic Text Using Natural Language processing Techniques", 2014[13].**

In this method the NLP (for Arabic text) techniques are utilized as a tool in order to increment the efficiency of the steganography. Each sentence in the cover text will be parsed in order to get its hiding method, so more than one hiding method is used in one text, and therefore the system competency is increased. This method depends on the grammar of the Arabic language to choose the method of hiding, i.e. each sentence must be parsed in order to obtain its grammar and according to this grammar the method of hiding will be chosen, so the security will be increased because multiple hiding methods will be utilized for one Arabic cover text. The B+ Tree is utilized to index the grammar in the lexicon.

- **Design of Proposed Arabic Text Steganography Approach Using Non Printed Character, 2016[14].**

In this method using specific Arabic Unicode characters with non-printed characters to hide the secret information. Using specified Arabic Unicode characters with non-printed characters to hide secret information provides complete similarity between cover text and stego text since these characters don't appear when written. This complete similarity gives as the ability to use the Arabic language features (from cover text to stego text) to provide information used as indications to determine secret keys between the sender and the receiver. Also B+ tree is used for dealing with the proposed database in order to provide speed and efficient access to the desired database contents.

VII. THE PROPOSED METHOD

The proposed information hiding system consists of two stages: sending and receiving stage. Sending stage will take from the user as input two texts (the English secret text and the Arabic text as the cover) and the output from this stage will be an Arabic stego text that will be used by the receiving stage to extract the original English secret text.

7.1 Sending Stage

This stage consists of main steps as in figure**2,** and algorithm**1** illustrate these steps.

Algorithm (1): Sending Stage
Input: Secret English text (**T**), Arabic cover text (**T1**).
Output: Stego text (**S**).
Begin
Step1: Normalize **T** by converting all its characters into lower case.
Step2: Call algorithm (**2**) that take **T** as input and return binary codes (**B**) and list of check bits (**List1**).
Step3: Call algorithm (**3**) that take **List1** as input and return non printed characters list **UL**.
Step4: Call algorithm (**4**) that takes **B**, **UL** and **T1** as input and returns stego text(**S**).
Step5: Return (**S**).
End.

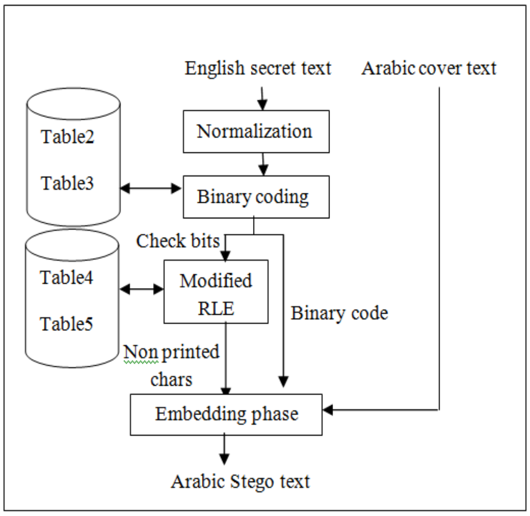

Figure (2): The main steps of the proposed sending stage

The normalization step converts the secret English text into lower case to be the input to the binary coding step that convert it into binary codes by using table2 and table3 and a list of check bits, such that any characters will be found in table2 will take one as a check bit otherwise it will take zero (see algorithm2). By using this coding method we can represent each character in the English secret text with six bits(five for binary coding and one for its check bit) instead of eight bits and this will give us a good ratio of unlooses compression.

Table (2): Characters that have one as a Check bit

character	' '	'a'	'b'	….	'"'	'.'	';'
5 bits	00000	00001	00010	….	11101	11110	11111

Table (3): Characters that have zero as a check bit

special character	'!'	'?'	'@'	…	'7'	'8'	'9'
5 bits	00000	00001	00010	…	11101	11110	11111

Algorithm (2): Binary Coding
Input: English text (**T**)**.**
Output: Binary code (**B**),
List of check bits (**List1**).
Begin
Step1: initialization
1.1 **List1**= []
1.2 **B**=""
Step2: While (**T**) ≠ "" do
Begin
2.1 Get the front character **C** from **T**
2.2 If **C** found in table (2) then
2.2.1 Get the corresponding binary code (**BC**) to **C** from **table2**
2.2.2 Add (**1)** to (**List1**).
Else
2.2.3 Get the corresponding binary code (**BC**) to **C** from **table3**
2.2.4 Add (**0**) to (**List1**).
2.3 add (**BC)** to (**B)**.
End while
Step3: Return (**B** and **List1**).
End.

After that we apply the proposed modified run length encoding (RLE) on the check bits list that result from the binary coding step, and this list always contains a consequence of ones with few zeros since each alphabetical letter (that appears frequently) found in table2 take one and the other special characters (that appear few times) will take zero, therefore (RLE) will be suitable for compression in this case (see algorithm3). In modified RLE we will depend on two databases one for zeros counters (as in table4) and the other one for ones counters (as in table5), such that each counter will be replaced by the ASCII code of a non-printed character that will be suitable for steganography purpose.

Table (4): Mapping of zero's counter

Counter	0	1	2	3	4	5	6	7	8	9
Unicode	2	1	4	6	8	20	21	9	3	5

Table (5): Mapping of one's counter

Counter	0	1	2	3	4	5	6	7	8	9
Unicode	22	24	23	19	17	15	16	11	12	14

Algorithm (3): Modified RLE
Input: List of check bits (**List1**).
Output: List of ASCII of none printed characters (**UL**).
Begin
Step1: initialization
UL= [], **i**=1
Step2: Compute length of **List1** to be **N**
Step3: while **i <= N** do
Begin
3.1 C=1
3.2 j=**i**+1
3.3 While **List1 [i]** =**List1[j]** do
3.3.1 C=**C**+1
3.3.2 j=**j**+1
End while
3.4 If **List1[i]** =0 then
3.4.1 for each digit (**D**) of **C** do
3.4.1.1 Use Table**4** to get the corresponding ASCII (**UN**) of **D**
3.4.1.2 Add **UN** to **UL**
Else
3.4.2 For each digit (**D**) of **C** do
3.4.2.1 Use Table**5** to get the corresponding ASCII (**UN**) of **D**
3.4.2.2 Add **UN** to **UL**
End if
3.5 i=j
End while
Step4: Return (**UL**).
End.

The last step is embedding phase that take the list of ASCII code of non-printed characters (that results from modified RLE), with the binary code (that result from binary coding), and take the Arabic cover text**.** Algorithm (4) illustrate this phase that use the spaces between the words of the

Arabic cover text in order to embed the ASCII code of non-printed characters, and use the Unicode character to embed binary code, such that if the code is one we embed a Unicode character according to the letter type (Unicode that has 158 ASCII for connected letter and Unicode that has 157 ASCII for isolated letter of Arabic cover text), and if the code is zero we leave the cover letter without any embedding. As an indication to the receiver to the end of binary code we embed 18 which is the ASCII code of non-printed character.

If we reach to end of cover and we still have a binary code or a non printed characters then we will add all the remaining non printed characters to the end of stego text, also for each one of binary code we will add the non printed character that have 13 ASCII code, and for each zero of binary code we will add the non printed character that have 25 ASCII code.

Algorithm (4): Embedding phase
Input: List of ASCII of non-printed characters **(UL),** binary code (**B**), Cover text **(T).**
Output: Stego Text **(S).**
Begin
Step1: initialization**:**
1.1 S="".
1.2 N=length of **UL.**
1.3 M=length of **B**.
1.4 L=length of **T.**
1.5 i=**j**=**k**=1.
Step2: while **j**<=**L** and (**i** <= **N** or **k** <= **M**) do /* While we not reach to the end of cover text and we still have binary code or non printed characters do*/
Begin
2.1 If **T[j]** ≠ ' ' then
Begin
2.1.1 Add **T[j]** to **S** /* we add the current cover character to the stego text */
2.1.2 If **k<=M** then /* if we still have a binary code */
Begin
2.1.2.1 If **B[k] =1** then
If **T[j]** is connected character then
Add the Unicode character that has 158 ASCII to **S**
Else
Add the Unicode character that has 157 ASCII to **S**
2.1.2.2 k=k+1
End if
Else
Add the non-printed character that has 18 ASCII to **S** /* the stop mark of binary code*/
End if
2.2 Else /* if T[j] = ' ' */
2.2.1 If **i**<=**N** then /* if we still have a non printed characters */
Begin
2.2.1.1 Add **UL[i]** to **S**
2.2.1.2 i=i+1
End if
2.2.2 Else
2.2.2.1 Add **T[j]** to **S** /* add the current space character */
2.3 j=j+1
End while
step3: if **j**>**L** and (**i**<**N** or **k**<**M**) then /* if we reach to end of cover and we still have binary code or non printed characters */
begin
3.1 while **i**<=**N** do /* while we have a non printed characters do */
Begin
3.1.1 Add **UL[i]** to **S** /* add the non printed character to the end of stego text*/
3.1.2 i=i+1
End
3.2 While **k<=M** do /* while we still have a binary code do*/
Begin
3.2.1 If **B[k] =1** then
Add the non-printed character that has 13 ASCII to **S**
Else Add the non-printed character that has 25 ASCII to **S**
3.2.2 k=k+1
End
End
Else /* we not reach to the cover text end yet */
3.3 While **j**<=**L** do
Begin
3.3.1 Add **T[j]** to **S** /* add the remaining cover text to the stego text */
3.3.2 j=j+1
End while
Step4: Return(**S**).
End.

7.2 Receiving Stage

This stage consists of main steps as in figure3. The input to this stage will be the Arabic stego text and the output will be the extracted original secret text, algorithm (5) illustrates these steps.

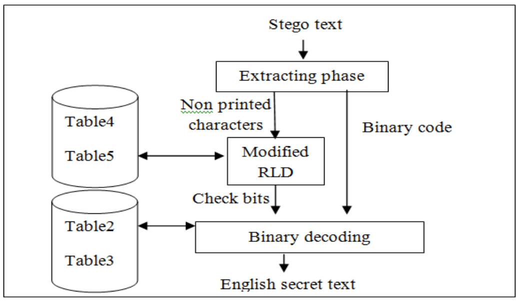

Figure (3): The main steps of the proposed receiving stage

Algorithm (5): Receiving Stage
Input: Stego text (**S**).
Output: Secret English text (**T**)
Begin
Step1: Call algorithm (**6**) that takes stegotext(**S**) as input and return **B**, **UL.**
Step2: Call algorithm (**7**) that take list of non printed characters **UL** as input and return list of check bits (**List1).**
Step3: Call algorithm (**8**) that take binary codes (**B**) and list of check bits (**List1**) as input and return **T**.
Step4: Return (**T**).
End.

The first stage is the extracting phase that extract the list of non-printed characters and the binary code as illustrated in algorithm6.

Algorithm (6): Extracting phase
Input: Stego Text **(S).**
Output: List of non-printed chars **(UL),** binary code (**B**)
Begin
Step1: initialization step
UL= []
B=""
N=length of **S.**
i=1, **j**=1, **k**=1
Flag=false
Step2: while **i**<=**N** do
Begin
If **S[i]** =18 then
Flag=true
Else
If **S[i]** =13 then
Begin
Add 1 to **B[k]**
k=**k**+1
End
Else
If **S[i]** =25 then
Begin
Add 0 to **B[k]**
k=**k**+1
End
If **S[i]** is non printed character then
Begin
Add **S[i]** to **UL[j]**
j=**j**+1
End
Else
If **S[i]** =158 or **S[i]** =157 then
Begin
Add 1 to **B[k]**
k=**k**+1
End
Else
If not flag then
Begin
Add **0** to **B[k]**
k=**k**+1
End
i=**i**+1
End //while
Step3: Return (**UL** and **B**).
End.

After that we will extract the list of check bits by applying algorithm (7), the input to this algorithm will be the list of non-printed characters that result from the previous extracting phrase.

Algorithm (7): Modified RLD
Input: List of non-printed characters (**UL**).
Output: List of check bits (**List1**).
Begin
Step1: initialization step
List1= []
i=1
N=length of **UL**
Step 2: while **i<=N** do
Begin
j=**i**
NC=0
M=1

While **UL[j]** is found in **table (4)** do
Begin
Get the corresponding value **C** of **UL[j]** from **table (4)**
NC=**NC**+**C*****M** // M is used if the counter is composed of more than one digit
j=**j**+1
M=**M***10
End //while
If **NC** >1 then
Begin
Add **NC** zeros to **List1**
NC=0
M=1
End //if
While **UL[j]** is found in **table (5)** do
Begin
Get the corresponding value **C** of **UL[j]** from **table5**
NC=**NC**+**C*****M**
j=**j**+1
M=M*10
End //while
If **NC** >1 then
Add **NC** ones to **List1**
i=**j**
End // While i<=N
Step3:-Return (**List1**).

End.

Figure (4) shows an example of using modified RLE in sending side and modify RLD in receiving side that take as input:

[0,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1] .

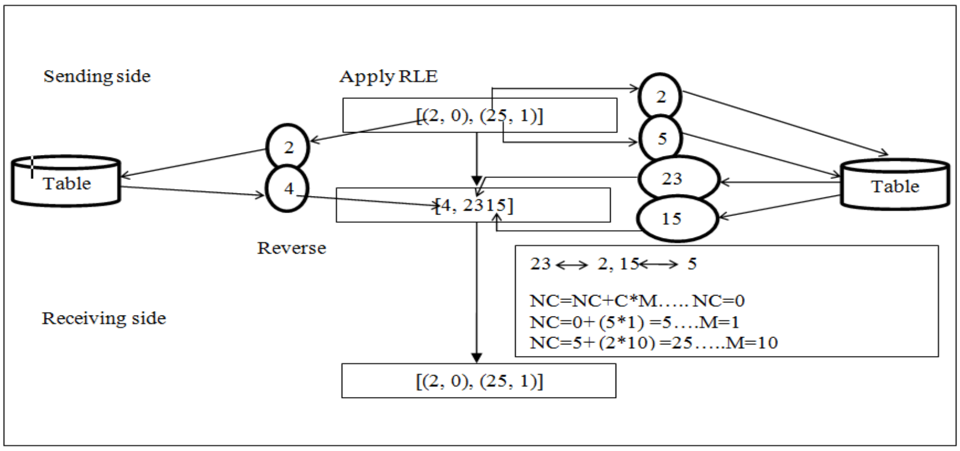

Figure (4): Example of Modify RLE and RLD

The last step will be the binary decoding to retrieve the English secret message (see algorithm8), the input to this algorithm will be the binary code that result from extracting phase and the check bits list that result from the previous modified run length decoding algorithm.

Algorithm (8): Binary Decoding
Input: Binary code (**B**), List of check bits (**List 1**).
Output: English text (**T**)
Begin
Step1: initialization
T = ""
i=1
N=length of **List1**
Step2: While **i**<=**N** do
Begin
2.1: Cut 5-bits **C** from **B**.
2.2 If **List1 [i]** =1 then
2.2.1 Get the corresponding character (**B1**) to **C** from **table (2)**
Else
2.2.2 Get the corresponding character (**B1**) to **C** from **table (3)**
2.3 Add (**B1**) to (**T**).
i=i+1
End // while
Step3: Return (**T**).
End.

VIII. IMPLEMENTATION OF THE PROPOSED METHOD

1) **Cover text:**

الحمد لله الذي علا في توحده ودنى في تفرده وجل في سلطانه
وعظم في أركانه وأحاط بكل شيء علماً وهو في مكانه وقهر جميع
الخلق بقدرته وبرهانه مجيداً لم يزل محموداً لا يزال بارئ
المسموكات وداحي المدحوات وجبار الأرضين والسماوات سبوح
قدوس رب الملائكة والروح متفضل على جميع براه متطول على
جميع من أنشاه يلحظ كل عين والعيون لا تراه كريم حليم ذو أناة قد
وسع كل شيء رحمته ومن عليهم بنعمته لا يعجل بانتقامه ولا يبادر
إليهم بما استحقوا من عذابه قد فهم السرائر وعلم الضمائر

2) **Secret Message:**

Steganography is defined as the science and art of hiding secret information

1. In coding step will convert secret message to stream of 5-bit and list of check bits:

B=10011101000010100111000010111001111001111001000001100000100011001000000 1…

List1=
[11111111111111111111111111111111111111111111111111111…11]

2. After apply the modified RLE: we have 76 of ones, so **UL**= [1116]
3. After apply hiding algorithm on cover text to embed non printed characters and binary codes:

الحمد لله الذي علا في توحده ودنى في تفرده وجل في سلطانه
وعظم في أركانه وأحاط بكل شيء علماً وهو في مكانه وقهر جميع
الخلق بقدرته وبرهانه مجيداً لم يزل محموداً لا يزال بارئ
المسموكات وداحي المدحوات وجبار الأرضين والسماوات سبوح
قدوس رب الملائكة والروح متفضل على جميع براه متطول على
جميع من أنشاه يلحظ كل عين والعيون لا تراه كريم حليم ذو أناة قد
وسع كل شيء رحمته ومن عليهم بنعمته لا يعجل بانتقامه ولا يبادر
إليهم بما استحقوا من عذابه قد فهم السرائر وعلم الضمائر

As we described there is a complete similarity between stegotext and cover text.

IX. PERFORMANCE ANALYSIS

Table (6) provides a comparison between the proposed modified RLE with the classical RLE. Also we will compute the similarity of the proposed method by using Jaro-Winkler Distance and compared with other related approaches (see table7). Also we will compute the capacity ratio of the proposed hiding method for different cover text file sizes (see table 8), and compared it with other methods (see table 9) by depending on equation (1).

Capacity ratio = (amount of hidden bytes) / (size of the cover text bytes)……… (1).

Table (6): A comparison between RLE and the proposed modified RLE

Modified RLE	**RLE**
Provides a good hiding tool	Not used for hiding purpose
Good for lossless compression with identical data	Good for lossless compression with identical data
Only the number of occurs (length) need to be represented	Character value and its number of occurs need to be represented
Not expansion tool with any data type	Expansion tool with un identical data
tool to embed block of data	Depend on information hiding method
Not depend on any table	Depend on two non-printed tables

Table (7): Jaro similarity score for the proposed and others approaches

Approach	**Jaro similarity score**	**Jaro Similarity score %**
AbdulRaheem approach [13]	0.9373	93.73%
Khan approach [19]	0.443	44.3%
Monika Agarwal approach [4]	0.95	95%
Sabrin Approach [14]	1.00	100%
Proposed method	1.0	100%

Table (8): Capacity Ratio of the Proposed Approach

Information Hiding Method	**Secret Message Sizes(byte)**	**Real Used Sizes of Cover(byte)**	**Hiding Capacity Ratio (byte/byte)**
Example(1)	76	912	8%
Example (2)	542	912	32%
Example (3)	822	912	90%
Example (4)	969	912	106%
Example (5)	986	912	108%

Table (9): Capacity ratio of the other approaches

Approach	**Average of capacity (byte/byte)%**
Kashidaa approach [15]	10.25%
Gutub and Fattani [11]	33.68%
Sabrin Approach [14]	89%
Shirali-Shaherza [16]	74.32%

X. CONCLUSION

In this section we will state the main conclusions:

1- Don't raise suspension because Complete similarity between cover text and stegotext..
2- Using more than one hiding methods in the same cover text provide a complexity that suitable for security purpose.
3- The proposed coding provides a compression ratio (reach to 30 %) and complexity that are suitable for security purpose.
4- The supposed modified RLE is more suitable for steganography purpose than RLE as shown in table6.

REFERENCES

[1] Aelphaeis Mangarae" *Steganography FAQ*", Copyright Zone-H.Org **2006**.

[2] M. K. Kaleem., " *An overview of various forms of linguistic steganography and their applications in protecting data*", Volume 3, No. 5, May **2012**.

[3] Ammar Odeh1and Khaled Elleithy," *Steganography In Arabic Text Using ZeroWidth And Kashidha Letters*", International Journal of Computer Science & Information Technology (IJCSIT) Vol 4, No 3, June **2012**.

[4] Monika Agarwal, "*TEXT STEGANOGRAPHIC APPROACHES: A COMPARISON*", International Journal of Network Security & Its Applications (IJNSA), Vol.5, No.1, January **2013**.

[5] Fabien A. P. Petitcolas, Ross J. Anderson and Markus G. Kuhn. "*Information Hiding – A Survey*", *Proceedings of the IEEE, special*

issue on protection of multimedia content, July **1999**, pp. 1062 – 1078.

[6] Joseph Gardiner, Shishir Nagaraja," *StegChat: A Synonym-Substitution Based Algorithm for Text Steganography*", Submitted in conformity with the requirements for the degree of MSc Computer Security School of Computer Science University of Birmingham,**2012**

[7] David Salomon, "*Data Compression The Complete Reference*", Fourth Edition, Computer Science Department, California State University, springer, **2007**.

[8] M.P.Bhuyan, V.Deka , S.Bordoloi, *Burrows Wheeler Data Compression And Secure Transmission*,Gauhati University, **2013**.

[9] Allen Wyatt, **2015**, *Displaying Non Printing Characters*, Available at:

[10] http://wordribbon.tips.net/t008879-DispalayingNonPrintingcharacters.html

[11] Micha, Kosmulski, **2004**, Introduction to Unicode, available at:

[12] http://www.Linux.com/archive/articles/39911

[13] Adnan A. G., Manal M. F.," *A Novel Arabic Text Steganography Method Using Letter Points and Extensions* "**,** World Academy of Science, Engineering and Technology International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:1, No:3, 2007.

[14] Akbas E.A., "*A New Text Steganography Method By Using Non Printing Unicode Characters*", Eng & Tech Journal, Vol.28, No.1, **2010**.

[15] AbdulRaheem A.A.," *Information Hiding in Arabic Text Using Natural Language Processing Techniques"*, M.Sc. thesis, Department of Computer Science ,The University of Technology, 2014.

[16] Suhad M.K.,Sabrin J.B.," *Design of proposed Arabic text Steganography approach using non printed character*", M.Sc. thesis, Department of Computer Science ,The University of Technology, **2016**.

[17] Adnan A.G., Wael A., and Abdulelah M., "*Improved Method of Arabic Text Steganography Using Extension Kashida Character*", Bahrain University Journal of information & communication Technology, December, 2010.

[18] Shahreza M. H. S., and Shahreza M. S., "***A New Approach to Persian/Arabic Text Steganography***" , In Proceedings of 5th IEEE/ACIS Int. Conf. on Computer and Information Science and 1st IEEE/ACIS Int. Workshop on Component-Based Software Engineering, Software Architecture and