King Fahd University of Petroleum & Minerals

Computer Science & Engineering

HW#1

Steganography:

The Idea and References

By:

Aleem Khalid Alvi

Yousef Salem Elarian

Submitted to Dr. Adnan A. Gutub

Computer Engineering Department

KFUPM

for the course of

Applied Cryptosystems: Techniques and Architectures

COE 509

March, 2007

Table of Contents

1. Preface

إن الحَمد لله. نحمده ونستعينه ونستغفره. ونعوذُ بِاللهِ من شرور أنفُسِنا وسيئّات أعمالنا. من يهده اللهُ فَلا مضل لَه. و من يضللْ فَلا هادى لَه. وأشهد أن لا إله إلا الله وحده لا شريك لَه وأشهد أن محمداً عَبْدُهُ ورسولُه.

Verily, all praises are due to Allah (S.W.T.), we praise HIM, seek HIS aid and beg for HIS forgiveness. We seek refuge from the Shaitan, from the evil of our own souls and the bad consequences from our deeds. Whomsoever, Allah (S.W.T.) decides to guide, no one can lead astray, and Whomsoever that Allah (S.W.T.) decides to lead astray no one can guide. I bear witness that none (i.e., GOD, deity, etc.) has the right to be worshipped but Allah (S.W.T.) and that Muhammad is the servant, Messenger of Allah.

2. Introduction and Motivation

Arabic diacritic marks decorate Arabic script to specify (short) vowels [1]. Those marks, shown in Figure 2-1, normally come over/beneath Arabic consonant characters. Arabic readers are trained to deduce these [2][3]. Just to feel such task, read the English sentence Figure 2-2 [4]. Vowels occur pretty frequently in languages. Particularly in Arabic, the nucleus of every syllable is a vowel [5]. Inside the computer, these are represented as characters [6].

The use of diacritics is an optional, not very common, practice in modern standard Arabic. However, when it comes to holly scripts, their use becomes the norm. Sameh *et al.* [7] have proposed to exploit these (somehow) redundant decorations to hide binary bits for steganography. In this paper, we extend their idea to the usage of multiple instances of the diacritic each mark to hide a message of arbitrary length; hence, increasing capacity.

Typically, in steganography, a tradeoff between security, capacity, and robustness manifests itself [8][9]. This one is not an exception. What makes the expense of security and robustness levels feasible, however, is that, in some scenarios, the new method is able to achieve arbitrary capacity, even from a single diacritic mark. Other tradeoffs will be discussed, as well.

Figure 2-1. Arabic diacritic marks.

Just to feel the task, read the following English sentence:
"jst t fl th tsk, rd th fllwng nglsh sntnc"

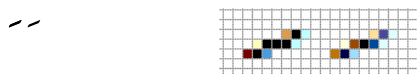
Figure 2-2. An English sentence and its non-vowelized version (reproduced in part from [4]).

3. Idea and Justification

The idea emerges from the way how computers display/print Arabic diacritic marks. When the code of a diacritic mark is encountered, the image of the corresponding stroke is rendered to the screen/printer without changing the location of the cursor. Such displaying without displacing leads to the possibility of typing multiple instances of a *almost* invisible extra-diacritics. A computer program aware of their presence and meaning, however, can easily detect and interpret them.

One extreme scenario of this method achieves a capacity of virtually infinity: The whole message can be hidden in a single diacritic mark by hitting (or generating) a number of extra-keystrokes equal to the binary number representing the message. Loss of robustness, compared to Owaidah *et al* [7] original 0-or-1 diacritics, manifests itself in the cases like printing out the document. While a printed, and maybe an OCRed, version can still carry the information in their method, such processing *almost* deprives the hidden message. Security of this scenario also may deteriorate. While on the code-level the original work of Owaidah *et al* [7] seems innocent, our code-file seems to have multiple instances of diacritics which might increase the file's size. On the other hand, our usage of diacritics is more consistent and might look exactly like the original cover-message, for bared eye. We can lessen the impact of this last security vulnerability by limiting the capacity, again, of each stroke to up to, say, three extra-strokes (i.e. three secret bits). This act will have a beneficial effect on robustness in the second scenario described next.

In the previous paragraph, we emphasize the word almost twice when discussing the robustness of our method: when qualifying the invisibility and the loss of information due to displaying/printing the multiple characters. The rationale for such act is a notice that, actually, the multiple type of a diacritic character *does* have an effect on the display. Although it doesn't displace the cursor, it does darken the displayed/printed diacritic stroke. Figure 2-1 shows the effect of a single and double Fathas on the rendered images. Note the darkness of the LHS ones. In conclusion, The idea is to carry *messages over diacritics*, instead of the normal case of having *diacritics over words*.



(a) (b)

Figure 3-1. A single and a double Fatha printed by MS-word in (a) and magnified as an image in (b).

This discovered fact is double folded. It exposes the security to decline for fear an of bear eye notice of the slight difference in levels of the darkness of the marks. The gain, however, is in terms of robustness. A trained eye can still be able to retrieve the hidden message after a printed version of the cover is produced. A program perhaps can manipulate the scanned page, as well.

4. Methods and Tools

The methodology to follow for the research is to generate multiple diacritics and test their appearance. These can be examined by zooming them and by converting them into images and analyzing the images. The tools for this methodology will consist in MS-Office® programs, namely, MS-notepad and MS-word for text generation, and MS-paint (plus the CImg.h library, if needed) for image examination. Google® will be used as a tool to find cover text.

5. Expected Results

A Java code is expected to accept the cover and hidden messages and output the stega-message. Besides, we can provide some statistics on the amount of usage of Arabic diacritics on the internet, according to the google® search engine. These figures would help supporting our premise of using the diacritics to as secure covers. In case of need, we can apply image processing analysis to demonstrate the change in gray levels between single- and multiple-diacritics.

We expect the program to function in one of two ways:

1. To output a hidden message of arbitrary length into any (possibly partially) diacriticised word. This way has emphasis on capacity. The cover message is of reasonable length, security is enhanced.
2. To output a balanced message of double and single diacriticised text. Here we sacrifice some capacity to gain robustness and security.

6. References

1. Amin A. Off-line Arabic character recognition: The state of the art. Pattern Recognition 1998; 31(5): 517-30.

2. Romeo-Pakker K, Miled H, Lecourtier Y. A new approach for Latin/Arabic character segmentation. IEEE 1995; 874-7.

3. Al-Anzi FS. Stochastic models for automatic diacritics generation of Arabic names. Computers and the Humanities 2004 38: 469-481.

4. A. Challenges in Arabic NLP. Sakhr Software Co. Arabic Language Resources (LR) and Evaluation: Status and Prospects. 1126-30.

5. El-Imam YA. Phonetization of Arabic: rules and algorithms. Computer Speech and Language 2004; 18: 339–73.

6. Abandah G, Khundakjie F. Issues Concerning Code System for Arabic Letters. Dirasat Engineering Sciences Journal, April 2004. 31 (1): 165-77.

7. Owaidah S, Shafei AbdelRahman, Aabed M. A new approach to steganography using diacritic marks. [In a class comment.]

8. Gutub AA. A novel Arabic text steganography method using letter points and extensions. [Still not published.]

9. Johnson NF, Jajodia S. Exploring Steganography: Seeing the Unseen. IEEE Computer, 1998. 31 (2):26-34.

7. Appendix A:

A Summary of "A New Approach to Persian/Arabic Text Steganography"

By: M. Hassan Shirali-Shahreza Mohammad Shirali-Shahreza

Assigned to: Aleem K Alvi.

In this paper, a new approach is described for Arabic text steganography. Text steganography is difficult because the structure of text a document is more visible and often less redundant than an image or a sound file. However, it's still used for the lesser memory requirement and more use on the net.

The present paper makes use of one of the characteristics of Persian and Arabic languages: the abundance of points on their characters to suggest a new feature coding method. The following algorithm was proposed by the authors:

```
Algorithm: /* Text steganography */
1. S = Compress (secret information);
2. C = Cover letter with p at the beginning
3. p = next pointed letter in cover;
4. b = Read next bit of S;
5. IF (b = 1) Shift the point on the concerned character a little upward;
6. Repeat steps from 3 to 5 till the end of S (if C ends first, error);
7. Changed_Randomly(The points of the remaining characters);
```

The bits are recovered by identifying all hidden bits in the character based on the place of points on the character. Decompression is then carried on.

Advantages	Disadvantages
1. Good capacity. 2. No much fear from OCRs in Arabic. 3. Information survives printing. 4. Information survives resizing.	1. Information lost with retyping. 2. Fixed frame (one font). 3. Scanned text may not survive.

The authors finally use several Iranian newspapers to prove the high capacity of their algorithm.

8. Appendix B:

A Summary of "A Novel Arabic Text Steganography Method Using Letter Points And Extensions"

By: Adnan Abdul-Aziz Gutub

Assigned to: Yousef S I Elarian.

This paper presents a new steganographic approach for Arabic script. It can be classified under steganography feature coding methods since it alters the way how a character/word looks like. The approach hides secret information by convening on a meaning to a kashida (the Arabic extension redundant character used for justifying or beautifying the text) after/before a pointed/bare character. The author uses the pointed letters with extension to represent one level of bits and the un-pointed letters with extension to hold the other state of it. This steganography technique is also found attractive to other languages having similar texts to Arabic such as Persian and Urdu.

The advantage of character feature methods, in general, is that they can hold a large quantity of secret information without making normal readers aware of the existence of such information in the text. Here, the author has sacrificed some of the capacity shown in a previous work for Persian and Arabic script to achieve better robustness. Such practice is implied by the nature of some Arabic characters that cannot have kashida's before/after them. A tradeoff like this one is common in steganography. For example, I would suggest achieving better capacity by applying kashidas alone (regardless of points). An extendable character which is extended would represent some level of binary information while a non-extended one would represent the opposite case. This time, maybe security would pay the bill.

Instructor Comments:

Your proposal of using diacritic is interesting. However, you need to figure a clear way to hide and then retrieve the hidden bits from the stego-file.
Study how can you hide bits in an already used stego file, it seems that the capacity will go very low. What is the cost if we want to have the same cover file for hiding data with same capacity?

You have to make a statistical study comparing different diacritics and their percentages affecting the capacity.
In general, your idea is very interesting; hopefully we can get your code running correctly.