RESEARCH PAPER

Parallel Execution of the Steganography using SFLA on the Android Platform

Manar Younis Ahmed   and Balqees Talal Hasan
Department of Computer Sciences, College of Computer Sciences and Mathematics, Mosul University, Mosul, Iraq.

ABSTRACT:
A single core processors have become part of the past, processors manufacturers have continued of producing multi_core processors for many devices such as smart phones and tablets. By this act ,they enabled developers to develop multi-threaded application to speed up the work and increase efficiency of their applications.

The goal of this research is to exploit the principles of parallel processing to construct integrated steganography application, which is running on the android platform. The research begins by providing a new suggested method for creating sketch images, as a cover for the secret data. The advanced encryption standard algorithm, which is provided by the Java Encryption Extension package, was used to encrypt the secret data and then the Shuffled Frog Leaping Algorithm was used to search for the best positions in the sketch image where the secret data can be embedded. It was developed here to work in parallel on quad-core processors. New embedding methods were proposed to embed the secret data based on the fitness of the best frog obtained from the SFLA . the embedding and extracting processes were done in parallel manner. Finally, this research provides the possibility of sending the stego image directly through techniques and some image-sending apps installed on the device .

The highest speed up obtained when applying the SFLA algorithm in parallel manner was (3.72) and the highest efficiency was (0.93) while the highest speed up in the embedding process was (3.3), and the highest efficiency was (0.82) and the highest speed up in the extracting process, was (2.21), and the highest efficiency was (0.55). The highest result of the proposed embedding way which reached it the PSNR measure was (1803.18 db) and the lowest measure was (0.0..1 db).

KEY WORDS: Steganography,Parallel Processing,SFLA,Sketche Images
DOI: http://dx.doi.org/10.21271/ZJPAS.31.s2.10.

INTRODUCTION :

With the rapid growth of the smart phones and tablets market, the issue of data security has migrated from computer platforms to those devices . One of the most important operating systems that operates these devices is the Android operating system (DigiFloor, 2013).

In order to increase the performance of the applications, the principles of parallel processing will be applied when designing and building them. Which has helped to apply complex algorithms to obtain more accurate and better results; which was difficult to apply on mobile phones due to their limited possibilities (Domeika, 2008).

Steganography is the art and science of hiding information. it is a process that involves hiding a message in an appropriate career. The selection of the carrier is an important issue, so that the carrier must be carefully selected. Once it has been used the carrier should not be used again and should be destroyed. A familiar
image should not be used so it is better for
the steganographer to create his / her own images
(Curran and Bailey , 2003). To accomplish that, in
this work non-photorealistic rendering (NPR)
has been used to convert the cover image into
sketch image.

NPR refers to the generation of images that
through technological means mimic images that
have been created by “the human hand”, thus the
final images will look like that made by an artists.
These types of images include paintings,
illustrations, sketches and drawings (Geng , 2010).

In this work , we intend to increase the
robustness of a steganography by hiding the secret
message in appropriate positions of the cover, to
achieve that shuffled leaping frog algorithm
(SLFA) is used to search about the best positions
of the image.

The SLFA is a memetic meta heuristic
that is designed to seek a global optimal solution
by performing a heuristic search (Elbeltagi, Hegazy,
and Grierson, 2007).

2. RELATED WORKS

2.1. In The Field of Data Hiding Based on The
Principles of Parallel Processing

1. In 2012, the researcher (Bucerzan and
Nagham) published a study to simulate a
parallel algorithm that is executed on multiple
processors to embed an image in a gray image
file and retrieve it using Matlab Parallel
Processing Tools (Bucerzan, Nagham 2012).

2. In 2013, ((Kika , and Greca) presented research
to hide a secret message in an image based on
the principles of parallel processing in multi-
core processors on computers.

3. In 2015, Zhelezov and Paraskevov suggests a
parallel algorithm for the implementation of the
LSB steganographic method. A program for
cluster computer system in OpenMPI
environment is implemented and an assessment
of the acceleration of the program depending
on the number of the computing cores that
implement the program is made.

2.2. In The Field of Sketch Image Generation

1. In 1999, the researchers (Mario and John)
presented a study in which they analyzed and
simulated the various tools used by the painter
in the drawing, such as pencils, paper, erasers
and synthesizers (Sousa and Buchanan, 1999).

2. In 2005, (Jin and Baoxin) published a paper in
which they proposed an algorithm for the
automatic generation of pencil drawings from
personal images (Zhou and Li ,2005).

3. In 2012, the researchers (Holger, Jan, and
Sven) presented a study in which they used a
proposed (Extended Difference of Gaussians)
to generate a sketch image that contained the
effect of straight lines and based on the concept
of the layer blending to form the final image
(Winnemolle , Kyprianidis , and Olsen , 2012).

2.3. In The Field of Information Hiding using
Smart Phones that Rely on The Android System

1. In 2013, the researchers (Geetanjali and Savita)
submitted a research to encrypt an image and
then embed it in another image using the least
significant bit algorithm (. Kshirsagar and
Kulkarni ,2013).

2. In 2013, the researchers (Bucerzan, Ratiu, and
Manolescu) presented a research paper in
which they developed an application to encode
and embed text or image files in an image
based on the least significant bit algorithm
(Bucerzan, Ratiu, and Manolescu , 2013).

3. In 2014, the researchers (Savithri and Sudha)
presented a research to encrypt an image based
on an chaotic system characterized by its
randomness, and then embedded in the cover
image (Savithri , and Sudha, 2014).

2.4. In The Field of Information Hiding Based
on The Shuffled Frog-Leaping Algorithm:

1. In 2013, the researchers (Maryam, Roneq and
Masoud) presented a research in which they used "shuffled frog leaping" algorithm to select
locations where secret data would be embedded
based on the LSB algorithm (Habibi , et. al ,
2013).

3. THE PROPOSED APPROACH
The proposed steganography system depends on the implementation of the following steps:

1. Generate the cover image by applying a suggested method for that, which includes converting any selected image to a sketch image.
2. Specify the type of the secret message to be sent, and if it is a text, it will be encrypted first using the advanced encryption standard.
3. Use the SFLA algorithm to search about the best locations in the generated sketch image and use them as locations to include the secret data.
4. Send a stego image via the techniques and some installed sending applications on the device such as Bluetooth, Wi-Fi and Messenger directly from our application.

3.1. Sketch Drawing Generation

Since the quality of tools and materials affect the artwork, the user can determine the color of the pencil and the papers, to make the result similar to sketch, the following steps will be followed to generate the cover image:

1. To generate the first and the most important image (layer), edges are extracted from the image using one of the edge detection filters. From the experimental results we find that Canny edge detection filter cannot be used to give the desired effect, since laplacian and sobel give good results. The input to this step is an original image which is selected either from gallery or from camera.
2. Hatching effect can be added to the edge-detected image, which is the use of fine, parallel lines drawn closely together, often rapidly drawn, to create the illusion of shade or texture in a drawing.
3. The color of the detected edges can be changed based on the color selected by the user. This operation is performed by applying image blending modes.
4. In the final step, we select the paper to put the sketch image on it. So after the user select the paper texture, blending mode is applied between the final image and the paper texture.
Figure (1) show the steps of a sketch (cover) image generation.

Figure (2) flowchart of parallel SFLA
3.2. Secret Message Encryption

Android SDK includes Java Crypto_graphy Extension (JCE) interfaces that provide easy access to common cryptographic operations. One of the most important algorithms provided by that package is the advanced encryption standard algorithm which we use to encrypt the secret text (JCA).

3.3 Parallel implementation of SFLA

In this application the shuffled frog leaping algorithm (SFLA) is used to select the optimal locations to embed secret bits into them in non sequential order. We applied the sequential SFLA algorithm in parallel manner to get benefits from multi_core processors of the mobile devices.

In this work, the secret information will be embedded at the blue level only. The changes made will be less sensitive to the human eye and will not be easily understood in contrast to the red and green levels where the sensitivity of the eye to changes is high and the change is significant.

The changes in the dark colors is less sensitive by the human eye, unlike light colors, any change will be felt. On this basis, (SFLA) will search the locations of the dark elements in the cover image to embed the secret information in them.

Steps of Executing SFLA in Parallel Manner:

After obtaining the cover image data, the SFLA algorithm will be applied in parallel manner by following the steps below:

1. Generate the initial population of frogs randomly.
2. Evaluate the fitness of each individual in the population.
3. Sort the population in a descending order, based on their fitness values.
4. Portions the population into four memeplexes and generate four threads for execution. Each core will be responsible for execution of one of the threads.
5. Perform the process of memetics evolution (local research) in each memplex simultaneously.
6. Shuffle all the memeplexes.

7. Check the stop condition if it is satisfied then the best frog is returned in the community, otherwise it will repeat again starting from step "3".

3.4. Text Embedding Algorithm:

Input: Sketch(cover) image and the plain text.

Output: Stego Image.

Step 1: Read the data of the cover image.

Step 2: Encrypt the secret text.

Step 3: Apply the SFLA algorithm in parallel manner, to retrieve the best frog, as described above.

Step 4: Embed the length of the secret text, depending on the value of the third bit that will be illustrated in the next step in the last locations of the cover image.

Step 5: Determine the method of embedding the secret information by checking the fitness value of the best frog obtained. If it is an even number, the information will be embedded later in the frog locations in the following manner: After extracting the blue byte from the indicated location, the embedding will be in either the first or second LSB. The embedding bit will depend on performing X-OR operation of the bits values of that byte from the third bit to the eighth bit. If the final result is zero, the embedding will be in the first LSB, and if the result is equal to one, the embedding will be in the second LSB.

If the fitness value is an odd number, the secret information will be embedded later in the following manner: After extracting the blue byte from the indicated location, the embedding will be in either the first or second LSB. The embedding bit will depend on the value of a Third bit, if the value is equal to zero, the embedding will be in the first LSB, and if its value is one, it will be embedded in the second LSB.
**Step 6:** Divide the locations of the best retrieved frog and the encrypted secret text into four parts.

**Step 7:** Generate four threads, each thread will be given to a specific core. So that each core will embed a certain part of the encrypted text in the assigned locations, and the embedding process done at the level of blue only in each location, and the four cores will do so at the same time.

**Step 8:** Collect the pixels from the four cores and create the stego image.

**Text retrieval algorithm**

**Input:** Stego image.

**Output:** Secret text.

**Step 1:** Read the data of the stego image.

**Step 2:** Retrieves the length of the embedded secret message, depending on the value of the third LSB, from the last locations in the stego image.

**Step 3:** Apply the SFLA algorithm in parallel manner, to retrieve the best frog, as described above.

**Step 4:** Determine the method of extracting the secret information by checking the fitness value of the best frog obtained. If it is an even number, the information will be extracted later from the frog locations in the following manner: After extracting the blue byte from the indicated location, the extracting will be either from the first or second LSB. The extracting bit will depend on the value of a third bit, if the value is equal to zero, the extracting will be from the first LSB, and if its value is one, it will be extracted from the second LSB.

If the fitness value is an odd number, the secret information will be extracted later in the following manner: After extracting the blue byte from the indicated location, the extracting will be either from the first or second LSB. The extracting bit will depend on performing X-OR operation of the bits value of that byte from the third bit to the eighth bit, if the final result is zero, the extracting will be from the first LSB, and if the result equal to one, the extracting will be from the second bit.

**Step 5:** Divide the locations of the best retrieved frog into four parts.

**Step 6:** Generate four threads, each thread will be given to a specific core. So that each core will extract an encrypted text from a certain part simultaneously with other three threads.

**Step 7:** Collect the extracted encrypted text.

**Step 8:** Decrypt the extracted text.

**3.5. Image Embedding Method**

**Input:** Sketch(cover) image and the secret image.

**Output:** Stego image.

**Step 1:** Read the data of the cover image and the secret image.

**Step 2:** Apply the SFLA algorithm in parallel manner, to retrieve the best frog, as described above.

**Step 3:** Embed the width and high of the secret image in the last locations of the cover image.

**Step 4:** Divide the locations of the best retrieved frog and the secret image into four parts.

**Step 5:** Generate four threads, each thread will be given to a specific core, So that each core will embed a certain part of the secret image in the assigned locations. Embedding the secret image data will be done in two bits of the blue and green colors from the retrieved pixel from those locations, and the identification of the two bits that will be used for embedding will
depend on the value of the fourth bit. If the value is equal to zero, the embedding will be done in the first and second LSB, and if the value is equal to one, the embedding will be done in the second and third LSB, and the four cores will perform the embedding process at one time.

Step 6: Collect the pixels from the four cores and create the stego image.

Image Extracting Method

Input: Stego image

Output: Secret image

Step 1: Read the data of the stego image.

Step 2: Retrieves the width and high of the secret from the last locations in the stego image.

Step 3: Apply the SFLA algorithm in parallel manner, to retrieve the best frog, as described above.

Step 4: Divide the locations of the best retrieved frog into four parts.

Step 5: Generate four threads for execution, as each core will be responsible for a particular thread. Thus, each core of the four processor cores will retrieve the secret image data from one of the four sections of the best frog locations granted to it and in a synchronous manner with the work of the other three cores, the retrieval of the secret image data will depend on retrieving value of two bits of the blue and green colors from the retrieved pixels from the frog locations, the identification of the two bits that their values will be extracted will depend on the value of the fourth LSB. If the value is equal to zero, the first and second LSBs will be retrieved, otherwise the second and third LSBs will be retrieved.

Step 6: Collect the extracted pixels of the secret image.

Step 7: Generate the secret image.

4. RESULTS

- The implementation of the searching for the best locations in the cover image in parallel manner decrease the time required to implement SFLA and thus increased the speed of implementation of the application and its efficiency.

- When the embedding and retrieval processes of the secret text were done in parallel, an increase in the speed and efficiency of the execution was achieved, due to a shorter execution time compared with its sequential execution.

- When the embedding and retrieval processes of the secret image were done in parallel, a relative increase in the speed and efficiency was achieved.

- The implementation of the SFLA through the proposed method has yielded good results in the inability to detect.

- The results of exposing the stego images to attacks demonstrated the ability of the sketch images, which were configured with the SFLA method to search for the embedding / retrieval of the secret data, to counteract the attack when it occurred as a result of obtaining acceptable results thereafter.

Tables (1) and (2) illustrate the results of embedding text in an image and image in an image depending on the proposed method.
Table (1) The results of embedding text in an image based on the SFLA

<table>
<thead>
<tr>
<th>Original Image</th>
<th>Cover Image</th>
<th>Size of a cover in Pixel</th>
<th>Length of the text in byte</th>
<th>Execution manner</th>
<th>Process</th>
<th>Time of SFLA execution in M.S.</th>
<th>Speedup</th>
<th>Efficiency</th>
<th>Times of embedding/extracting processes in M.S.</th>
<th>Speedup</th>
<th>Efficiency</th>
<th>Before PSNR in db</th>
<th>Attack MSE</th>
<th>After PSNR in db</th>
<th>Attack MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential Embedding</td>
<td>0.648</td>
<td>129</td>
<td>338</td>
<td>567</td>
<td>129</td>
<td>6.42</td>
<td>0.5</td>
<td>66</td>
<td>39</td>
<td>81.307</td>
<td>0.0004</td>
<td>57.0417</td>
<td>0.1284</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Embedding</td>
<td>337</td>
<td>3.12</td>
<td>0.78</td>
<td>20</td>
<td>3.3</td>
<td>0.82</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Extracting</td>
<td>388</td>
<td>1.13</td>
<td>0.64</td>
<td>26</td>
<td>0.37</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential Embedding</td>
<td>0.66</td>
<td>430</td>
<td>338</td>
<td>567</td>
<td>430</td>
<td>7.48</td>
<td>0.49</td>
<td>133</td>
<td>10</td>
<td>73.140</td>
<td>0.33</td>
<td>57.249</td>
<td>0.43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Embedding</td>
<td>22.586</td>
<td>3.13</td>
<td>0.78</td>
<td>80</td>
<td>1.70</td>
<td>0.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Extracting</td>
<td>177.588</td>
<td>0.72</td>
<td>0.93</td>
<td>96</td>
<td>1.57</td>
<td>0.39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential Embedding</td>
<td>0.71</td>
<td>430</td>
<td>338</td>
<td>567</td>
<td>430</td>
<td>6.42</td>
<td>0.5</td>
<td>136</td>
<td>10</td>
<td>78.246</td>
<td>0.33</td>
<td>57.249</td>
<td>0.43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Embedding</td>
<td>64.403</td>
<td>3.44</td>
<td>0.86</td>
<td>105</td>
<td>1.45</td>
<td>0.36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Extracting</td>
<td>40.798</td>
<td>0.6</td>
<td>0.90</td>
<td>131</td>
<td>0.29</td>
<td>1.17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential Embedding</td>
<td>0.7</td>
<td>430</td>
<td>338</td>
<td>567</td>
<td>430</td>
<td>6.42</td>
<td>0.5</td>
<td>136</td>
<td>10</td>
<td>78.246</td>
<td>0.33</td>
<td>57.249</td>
<td>0.43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Embedding</td>
<td>22.586</td>
<td>3.13</td>
<td>0.78</td>
<td>80</td>
<td>1.70</td>
<td>0.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel Extracting</td>
<td>177.588</td>
<td>0.72</td>
<td>0.93</td>
<td>96</td>
<td>1.57</td>
<td>0.39</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table (2) The results of embedding image in an image

<table>
<thead>
<tr>
<th>Original Image</th>
<th>Cover Image</th>
<th>Size of a cover in Pixel</th>
<th>Size of a secret image</th>
<th>Execution manner</th>
<th>Process</th>
<th>Time of SFLA execution in M.S.</th>
<th>Speedup</th>
<th>Efficiency</th>
<th>Times of embedding extracting processes in M.S.</th>
<th>Speedup</th>
<th>Efficiency</th>
<th>Before Attack</th>
<th>After Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Original Image" /></td>
<td><img src="image2" alt="Cover Image" /></td>
<td>538 * 736</td>
<td>224 * 300</td>
<td>Sequential</td>
<td>Embedding</td>
<td>16369</td>
<td>166</td>
<td>82</td>
<td></td>
<td></td>
<td></td>
<td>24.473</td>
<td>9.2237</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Extracting</td>
<td>15935</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parallel</td>
<td>Embedding</td>
<td>6042</td>
<td>2.70</td>
<td>0.067</td>
<td>130</td>
<td>1.27</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Extracting</td>
<td>6155</td>
<td>2.58</td>
<td>0.64</td>
<td>76</td>
<td>1.07</td>
<td>0.26</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image3" alt="Original Image" /></td>
<td><img src="image4" alt="Cover Image" /></td>
<td>828 * 541</td>
<td>304 * 480</td>
<td>Sequential</td>
<td>Embedding</td>
<td>42824</td>
<td>92</td>
<td>124</td>
<td></td>
<td></td>
<td></td>
<td>67.349</td>
<td>0.126</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Extracting</td>
<td>43345</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parallel</td>
<td>Embedding</td>
<td>14042</td>
<td>3.04</td>
<td>0.76</td>
<td>86</td>
<td>1.06</td>
<td>0.26</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Extracting</td>
<td>14841</td>
<td>2.92</td>
<td>0.73</td>
<td>74</td>
<td>1.67</td>
<td>0.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image5" alt="Original Image" /></td>
<td><img src="image6" alt="Cover Image" /></td>
<td>750 * 600</td>
<td>320 * 680</td>
<td>Sequential</td>
<td>Embedding</td>
<td>20158</td>
<td>169</td>
<td>118</td>
<td></td>
<td></td>
<td></td>
<td>68.942</td>
<td>0.103</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Extracting</td>
<td>20337</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parallel</td>
<td>Embedding</td>
<td>8739</td>
<td>2.30</td>
<td>0.57</td>
<td>131</td>
<td>1.29</td>
<td>0.32</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Extracting</td>
<td>9049</td>
<td>2.24</td>
<td>0.56</td>
<td>116</td>
<td>1.01</td>
<td>0.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td><img src="image7" alt="Original Image" /></td>
<td><img src="image8" alt="Cover Image" /></td>
<td>826 * 741</td>
<td>340 * 430</td>
<td>Sequential</td>
<td>Embedding</td>
<td>386255</td>
<td>345</td>
<td>466</td>
<td></td>
<td></td>
<td></td>
<td>79.122</td>
<td>0.2125</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Extracting</td>
<td>508127</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Parallel</td>
<td>Embedding</td>
<td>216165</td>
<td>1.78</td>
<td>0.44</td>
<td>213</td>
<td>1.61</td>
<td>0.40</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Extracting</td>
<td>436099</td>
<td>1.16</td>
<td>0.29</td>
<td>210</td>
<td>2.21</td>
<td>0.55</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5. CONCLUSIONS

By applying an innovative method for cover image generation and the implementation of the parallel processing principles when constructing the proposed applications, and applying the embedding/extracting methods proposed in this work, and through the results reached, the following conclusions were reached:

- The proposed way to create the cover image, which depends on converting any image into a sketch image, provides multiple options to the user, and great ease and speed, so the nature of the generated sketch images will exclude the suspicion of the existence of the secret data contained within them.

- Using parallel processing principles when building applications on tablets and Android smart phones allowed us to implement complex algorithms that are difficult to implement sequentially. The SFLA algorithm is implemented using parallel processing principles to look for the best locations to embed the secret data, thereby increasing the speed and efficiency of the application; as a result of decreasing the execution time of the algorithm compared to its serial execution.

- Using the SFLA to search and use of dark pixels in the cover image, gave good results in the inability to detect the existence of embedded secret data.

- The combination of cryptographic and information hiding technologies gave greater security to the embedded data, so even if the intruder could detect the hiding, he would need to break the text code.

6. REFERENCES


