ABSTRACT: LSB is well known method for steganography. There are several changes suggested by different researchers time to time on LSB and steganography. Most of the methods are by using encryption and in decryption of the message in different ways. This paper proposes the Java APIs for encoding and decoding of messages which is to be staged.

Keywords: Steganography, image based steganography, Efficiency consideration in LSB method, image encoding algorithm, image decoding algorithm.

I. INTRODUCTION

The proposed method is using very compact and effective APIs of Java for implementation and understanding of encryption and decryption which is given in java.crypto package.

Efficiency Considerations

Which working with the concept of LSB, three main issues which needs to be covered for efficiency and security of message which is hidden in the image.

- Storing length of message: As per most of the references, the length is stored in first 31 pixels and data is stored in next each pixel. This is the general idea. So a malicious user can retrieve the first step of message by first 31 pixels.

- Storing message: After the length, the data is being stored in each pixel which can be easily retrieved by any malicious user. The popular method of the steganography method is LSB. And the beauty of encryption and decryption which is given in java.crypto package.

A lots of researchers have worked on concept of LSB. Some of them have designed good algorithm. But a very few number of researchers have taken it to implementation level. Our focus of the work was to implement one of the best models, in which the data can be moved in secure and safe way. We tried to study various language specifications. Java has implemented one of the best security model so we used java security for designing the tool.

- Encoding messages: The first concern of study was to make message more and more secure. Any malicious user can retrieve the data easily if the structure of storage of message is known. So one more thought was to given on security of data/message.
We found that data should be stored in LSB after encryption so that this will be next layer of security. If working with java, java provides a good set of classes in package javax.crypto for encryption and decryption.

In the modified method following changes can be proposed:

- For better image after embedding of message, we used alternative pixel for storing the message. If each message is being stored in each pixel, the image quality degrades so we stored each bit of message in even or odd number of pixel. Like first bit will be stored in 12th pixel and then next bit to 14th pixel and so on.

- For security of data, length is stored in some random pixel in the image. As per given in most of the standards, length is being stored in first 32 pixels which can easily be retrieved by any of the person for misuse. So we have used concepts that length is to be stored in any of the arbitrary location which is known only to sender and receiver. Agreed upon protocol will work and embed/decode the message accordingly

- Also for security of data, message can be encrypted by using java cryptography APIs which are available in javax.crypto package. So that the retrieval of exact message will be more tougher. The same is done by reverse at receiver end.

Comparison of Standard LSB method and Proposed Method

<table>
<thead>
<tr>
<th>Standard LSB Method</th>
<th>Our LSB Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message length is stored in first 31 bytes which is common and not safe.</td>
<td>Message length is stored in any arbitrary location.</td>
</tr>
<tr>
<td>Store 3 byte information per pixel</td>
<td>Store 1 byte information per pixel which will improve the result</td>
</tr>
<tr>
<td>Stego image looks almost same as the cover image.</td>
<td>Stego image looks same as the cover image, Better than Standard LSB.</td>
</tr>
<tr>
<td>The location of message in pixels are same</td>
<td>The location can be defined on the basis size of message to be embedded in the image</td>
</tr>
</tbody>
</table>

Image Encoding Algorithm in proposed method

Inputs: Hidden Message, Stego key and Cover Image
Output: Stego Image
Procedure
Step 1 calculate length of message and hide length in any 32-bit.
Step 2 extract the characters of Message.
Step 3 Encrypt message by using javax.crypto APIs.
Step 4 extract characters from stegokey.
Step 5 calculate pixel value from Cover Image.
Step 6 select LSB bit from pixel and select stego key characters and put into Image pixel.
Step 7 insert characters of message in pixels of Image.
Step 8 repeat step 6 until characters has been embedded.
Step 9 repeat step from 2 to 7 message length.
Step 10 output in Stego image.

Image Decoding Algorithm in proposed method

**Inputs:** Stego Image and Stego key

**Output:** Cover Image and Message

**Procedure**
Step 1 calculate length of message from given 32-bit pixel.
Step 2 extract pixels from Stego Image.
Step 3 start from 32 pixel and extract stego key characters from first component of the pixels.
Step 4 If this extracted key matches with the key entered by the receiver, then follow Step 5, otherwise terminate the program.
Step 5 If the key is correct, then go to next pixels and extract secret message characters from first component of next pixels. Follow Step 5 till up to terminating symbol, otherwise follow step 6.
Step 6 extract Characters from Stego Image.
Step 7 extract special bit from Characters.
Step 8 obtain message by decrypting using javax.crypto classes and Cover Image.

REFERENCES
• BassamJamilMohd, Saed Abed and Thaier Al-Hayajneh, Computer Engineering Department Hashemite University, Zarqa, Jordan Sahel Alounneh, Computer Engineering Department, German-Jordan University, Amman, Jordan, “FPGA Hardware of the LSB Steganography Method” IEEE 2012.


• Mehdi Kharrazi, Husrev T. Sencar, and Nasir Memon, Image Steganography and: Concepts and Practice”, Department of Electrical and Computer Engineering Department of Computer and Information Science Polytechnic University, Brooklyn, NY 11201, USA.


• Java.oracle.com