IMAGE STEGANOGRAPHY

Mrs.M.KIRUBA DEVI,SIVAKUMAR.P ,SIVASELVAN.S ,VASANTHAKUMAR.S ,UDHYA KUMAR.V

Department of Information Technology, Batchelor of Technology, Sri Shakthi Institute of Engineering and Technology(Autonomous),Coimbatore-641062

ABSTRACT

Image steganography is a method of concealing confidential information within digital photos such that its presence is not obvious. In this project, the Least Significant Bit (LSB) algorithm in the Java programming language is used to implement image steganography. To incorporate secret data, like text, the LSB technique changes the least significant portions of an image's pixel values without appreciably changing the original image.Message fragments are seamlessly inserted by adjusting each pixel's red, green, and blue (RGB) color components. The secret message is accurately retrieved during decoding by using a preset delimiter to indicate its end. Because of the method's high imperceptibility and little distortion, the stego image looks exactly like the original.Java's BufferedImage and ImageIO classes are used to process images in the system's implementation. This project illustrates a straightforward but efficient method of image-based data concealment for secure communication that can be used for low-level security, research, and instructional purposes.

INTRODUCTION

To guarantee secure and private communication, secret data can be hidden inside digital photographs using a technique called image steganography. Steganography conceals the existence of a message, as contrast to encryption, which safeguards its content. The Least Significant Bit (LSB) algorithm is one of the most straightforward and popular methods in image steganography.In order to embed data without appreciably altering the image itself, the LSB method modifies the least significant bits of an image's pixels. A color image's pixels are made up of red, green, and blue (RGB) components, each of which is represented by eight bits. It is possible to discreetly store fragments of a secret message by changing the final bit of each component.Java offers a strong foundation for LSB steganography implementation through classes such as ImageIO and BufferedImage, which provide effective image data manipulation. Reading the image, bit-by-bit encoding the message into pixel values, and saving the altered image to a file are all steps in the implementation process.The image is scanned, and the LSBs are read to recreate the original data in order to recover the concealed message. To mark the conclusion of a message, a delimiter is frequently employed. PNG and BMP are the lossless picture formats that work well with this technique.Java's LSB method is easy to use, effective, and appropriate for applications that need minimal data confidentiality and stealth.

OBJECTIVE

This project's primary goal is to use Java's Least Significant Bit (LSB) technique to construct image steganography, which allows for safe and undetectable data hiding inside digital photos. This method makes use of the capability to alter an image's least important pixel values without producing appreciable visual changes. The project's goal is to create a Java-based program that uses bitwise operations and Java's image processing tools, like BufferedImage and ImageIO, to embed and extract hidden messages from photos. For accurate decoding, the tool will use a delimiter to identify the end of the message and support encoding textual data into the RGB values of image pixels. In order to maintain data integrity, it will emphasize the use of lossless picture formats such as PNG or BMP.In order to make the notion understandable to novices and students, the project also aims to provide a straightforward and user-friendly interface for carrying out steganographic procedures. By accomplishing these objectives, the project lays the foundation for more sophisticated steganographic systems while showcasing how Java may be used for secure communication through digital media.

LITERATURE REVIEW

The technique of image steganography, which adds a layer of security through obscurity, has been extensively researched as a way to hide data within digital photos. The Least Significant Bit (LSB) algorithm is one of the most popular and straightforward methods among the many others. According to research, LSB-based techniques are very successful at concealing text or binary data in lossless picture formats like PNG and BMP because they maintain the concealed information without causing the image to change noticeably. According to research by Johnson and Jajodia (1998), the LSB approach is perfect for real-time applications since it has a large embedding capacity and minimal processing complexity. Improvements to LSB have been investigated in more recent studies to increase resilience against attacks like compression and noise addition.LSB has been successfully used to hide and retrieve communications with little distortion in previous academic project implementations. To improve data security during retrieval, a number of research also suggest using encryption or delimiters. Although LSB is vulnerable to some types of attacks and distortions, its effectiveness and ease of use make it a solid choice for simple steganography applications. According to the studied literature, Java-based LSB steganography is still a dependable and instructive method for safely concealing data as long as picture data is handled carefully and the right format is chosen.

METHODOLOGY

This methodology describes the methodical process utilized to use Java to implement image steganography. The project's foundation is the Least Significant Bit (LSB) technology, which is used to embed and retrieve hidden information from digital images.1. Analysis of RequirementsDetermine the steganography system's objectives, such as safe data concealment and an intuitive user interface.Choose which file types (text, image, etc.) to conceal.Select the format for the cover image (such as PNG or BMP because of lossless compression).

2. Programming Language for the Technology Stack: Java SE (Java Standard Edition)IDE: Eclipse/IntelliJ IDEA Libraries: Java AWT & Swing for GUI, javax.imageio for image processing3. Design input: hidden file or message and cover imageStego image as the output (with a hidden message embedded)Modules:The cover image is loaded and verified by the image loader.The encoder incorporates the secret message into the picture.Decoder: Takes the secret message out of the Stego picture.GUI Interface: For communication with users4. Steps for Implementation The process of embedding (encoding)Create a binary stream out of the secret message.BufferedImage reads the image pixel by pixel.5. Validation and TestingExperiment with various message lengths and image sizes.Make sure the stego image maintains its visual quality.Verify the extracted message's accuracy.

EXISTING METHODS

**1.** The 1-LSB Method of Basic LSB SubstitutionThis is the most basic type of LSB steganography, in which the secret message is substituted for the least significant bit of each pixel's RGB value.Qualities:Simple to put into practice.low pixel-by-pixel embedding capacity (1 bit per color channel).minimal distortion of the image.Java example: Read each pixel, change the LSB of R/G/B, and then write it back using BufferedImage.2. The n-LSB Method, or Multi-LSB SubstitutionThis technique increases capacity by changing several least significant bits (such as two or four bits) of each color channel rather than just one LSB.Qualities:increased capacity for embedding.If too many bits are changed, the distortion will be a little more obvious.Ideal for more substantial messages.3. LSB Embedding Based on the EdgeThis technique inserts data into pixels that are near edges or in highly contrasted regions. Changes in these areas are less likely to be visually apparent.Qualities:safer than LSB uniforms.lessens the possibility of being discovered.requires edge identification prior to embedding, such as through the use of the Sobel operator.Notes about Java Implementation:To find edges, apply image processing methods (such as convolution filters).4. Encrypted LSB Steganography Description: To add an extra degree of protection, the secret message is encrypted using a symmetric key (such as AES or a straightforward XOR cipher) prior to embedding.Qualities:Increased privacy.Content is safeguarded even if LSB bits are made public.After extraction, a decryption step is necessary.Java Flow:Message encryption → Binary conversion → Use LSB to embed.While being extracted: Extract the bits → To obtain the original message, decrypt it.

DISADVANTAGES

1.Image processing techniques including compression, resizing, and filtering can readily damage or delete the hidden message using LSB steganography.2.Because the size and resolution of the image determine how much data can be inserted, it has a limited hiding capacity and is not appropriate for concealing huge files.

3.Because of its absence of built-in protection, a password or encryption key is not required to extract the concealed message if one knows the technique.4.Larger lossless picture formats like BMP or PNG are the only ones that LSB steganography can work with since compression changes pixel values and erases the concealed data in lossy image formats like JPEG.

PROPOSED SYSTEM

The suggested system is a Java-based picture steganography application that conceals confidential data in digital photos by using the Least Significant Bit (LSB) technique. In order to guarantee that the concealed data is preserved, it is made to function with lossless image formats like PNG and BMP. The system's intuitive graphical user interface, created using Java Swing, makes it easy for users to generate stego images, input secret messages, and import cover images.The secret message is transformed into binary during the embedding procedure and concealed in the image pixels' least significant RGB color values. A distinct delimiter is used to indicate the conclusion of the message, and the message length is recorded at the start of the data to guarantee precise extraction. The concealed message is recovered and reconstructed by the system's decoding module. Unauthorized access is prevented by a password-protection mechanism, and the buried data is further secured by optional encryption (such AES or XOR). A trade-off between hiding capacity and image quality is made possible by the application's adjustable LSB depth. Multilingual text is made possible with Unicode support, and input validation makes that the message fits inside the image's bounds. A logging mechanism to record user actions, integrity checks while decoding, and an image preview are additional features.Operating on Windows, Linux, and macOS, the platform-neutral system is built with future scalability in mind, including possible extensions for more media types or cloud-based storage.

SYSTEM REQUIREMENTS:

Software Requirements:

HTML

CSS JAVA

# References:

* 1.  Johnson, N. F., & Katzenbeisser, S. (2000). *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House.
  2.  Provos, N., & Honeyman, P. (2003). *Hide and Seek: An Introduction to Steganography*. IEEE Security & Privacy, 1(3), 32–44.
  3.  Kaur, P., Singh, A., & Bansal, R. (2015). *A Review of Image Steganography Techniques*. *International Journal of Advanced Research in Computer Science and Software Engineering*, 5(4).