International Journal of Science, Strategic Management and Technology
Volume **02** Issue **11** November-2026 | ISSN: **3108-1762 (Online)** | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

Quantum-Resistant Cyber Security Apparatus

Pradeep Gowda
Head of the Department
Department of Computer Science and Engineering
Sri Revana Siddeshwara Institute of Technology, Chikkajala, Bengaluru, Karnataka, India – 562157

Ushashree B N
Assistant Professor
Department of Computer Science and Engineering
Sri Revana Siddeshwara Institute of Technology, Chikkajala, Bengaluru, Karnataka, India – 562157
ushashreebn@gmail.com

Arunkumar
Department of Computer Science and Engineering
Sri Revana Siddeshwara Institute of Technology, Chikkajala, Bengaluru, Karnataka, India – 562157
arungoudachannashetty@gmail.com

Siddanagouda N Patil
Department of Computer Science and Engineering
Sri Revana Siddeshwara Institute of Technology, Chikkajala, Bengaluru, Karnataka, India – 562157
Siddanagoudapatil979@gmail.com

Sushma B R
Department of Computer Science and Engineering
Sri Revana Siddeshwara Institute of Technology, Chikkajala, Bengaluru, Karnataka, India – 562157
brsushma70@gmail.com

Srushti G N
Department of Computer Science and Engineering
Sri Revana Siddeshwara Institute of Technology, Chikkajala, Bengaluru, Karnataka, India – 562157
srushtign9@gmail.com

https://doi.org/10.55041/ijsmt.v2i11.025

Cite this Article: Gowda, P., N, U. B., Arunkumar, , Patil, S. N., R, S. B. & N, S. G. (2026). Quantum-Resistant Cyber Security Apparatus. International Journal of Science, Strategic Management and Technology, *02*(11), 1-9. https://doi.org/10.55041/ijsmt.v2i11.025

License: This article is published under the Creative Commons Attribution 4.0 International License (CC BY 4.0), permitting use, distribution, and reproduction in any medium, provided the original author(s) and source are properly credited.

Abstract— We present a host based local application that integrates message encryption with image steganography. The user inputs a secret message, a cover image and a key using a browser interface. The Flask application orchestrates cryptographic processing, image handling and least significant bit (LSB) embedding. The encrypted payload is represented in bits and embedded in the selected pixel values of an image to create a stego image. During recovery, the embedded payload is extracted by the application and an attempt is made to decrypt it with the appropriate key. The project report details a Python implementation with Flask, Pillow, NumPy and SQLite. Crucially, the implementation chapter states that the cryptographic functions use AES via PyCryptodome in ECB mode, while other parts of the report refer to the algorithm as SM4.This paper therefore assumes that AES-ECB is as implemented in the code discussion and requires algorithm verification as a pre-publication check. The report provided is not a verified measured results report. No quantitative performance or security claims are made. The combination provides message confidentiality through encryption and concealment through steganography . But the combination itself is not a post-quantum cryptographic system .

I. INTRODUCTION

Communicating digitally makes it easy to send information quickly.. When the information is sensitive it can be at risk if not protected properly. If messages are. Saved without strong security people who are not supposed to see them might still access them. Cryptography helps by turning text into something called ciphertext. This ciphertext looks like data and can only be changed back to readable text with the right key.

Steganography works in a way. It does not try to hide the meaning of the message. Instead hides the message itself. It does this by placing data inside something that looks normal like an image. To the eye the image looks unchanged.. The hidden message is there and only someone who knows how to look for it can find it.

When both methods are used together they give two levels of protection. First encryption makes sure the message content is unreadable to anyone without the key. Then steganography hides the fact that a secret message exists all. This combination makes it harder for someone to even know there is a message.

The project is about a web application that runs on a computer. It lets a user put encrypted text into an image and later get it back. The app runs locally—no need for servers or hardware. The user interface is a web page in a browser. That interface talks to a Python Flask server in the background. For image work the app uses Pillow and NumPy.

Data like logs and file metadata are stored in SQLite. It can work on Windows or Linux.

The name of the project includes the phrase ‘ resistant’.. The actual implementation does not use any standard post-quantum algorithms. There is no use of establishment or digital signature methods that are designed to be secure against quantum computers. So the security of the system is based on encryption and least significant bit (LSB) steganography. This setup does not provide post-quantum security. The name might be misleading. The system should not be used as proof that it's resistant, to quantum attacks.

II. RELATED WORK

Image steganography has been studied as a way to hide information inside pictures. Spatial-domain methods change pixel values directly. LSB substitution is a spatial-domain approach where message bits replace the lowest bits of chosen pixel channels. Because only low-order bits are changed the visual difference may be hard to see under conditions. I find image steganography to be a technique. However statistical analysis can expose embedding patterns and common image operations such as resizing, cropping or lossy compression can harm data.

Cryptographic protection and image steganography work together. Encryption alone does not hide the fact that ciphertext is being exchanged. Image steganography alone does not necessarily protect the meaning of a message once it is taken out. I believe that encryption and image steganography should be used together. A combined design first encrypts the payload. Then hides the ciphertext in a carrier image. The receiver extracts the payload and decrypts it using the key.

The project report lists work on image steganography, steganalysis and LSB detection. The full bibliographic details, for entries are missing in the supplied report excerpts. Those references must be checked against the publications and completed before submission; unverified author names, titles, venues or page numbers should not be made up. I recommend checking all references.

III. PROBLEM STATEMENT

A message sent as text can show its content to someone who is not supposed to see it. Sending encrypted text can keep the content safe. It might still look like it is encrypted. A useful project for a student is needed to show a process where a secret message is encrypted and then hidden inside a picture. There should also be a way to get the message back out using the key and a picture that has not been changed.

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

International Journal of Science, Strategic Management and Technology
Volume 02 Issue 11 November-2026 | ISSN: 3108-1762 (Online) | Impact Factor: 3.8
An International, Peer-Reviewed, Open Access Scholarly Journal Indexed in recognized academic databases

The system needs to have a web page.It should let people put in a message and a picture. Then it should work on the message create a picture, with the message inside and get the original message back when the right key and the same picture are used. The way it works and what it can't do should be explained clearly. It should not say it is secure if that has not been tested or proven.

IV. OBJECTIVES

1. To create a web app that hides text inside pictures I plan to use a simple browser interface.
2. To lock the text before putting it into the picture I will use encryption.
3. To use the significant bit method for adding and taking out data I will write clear code.
4. To recover and unlock the hidden data with the key I will build a secure extraction tool.
5. To use a Flask backend that is split into modules and image tools I will keep code clean.
6. To keep logs and data about the app in an SQLite database I will record actions.
7. To write down limits and suggest steps, for stronger encryption I will note all caveats.

V. PROPOSED SYSTEM

The proposed application is a host-based secure-steganography system. A user uses a web browser to talk to the system. The frontend, made with HTML, CSS and JavaScript sends requests, to the Flask server. The backend checks the request and ties together the image-processing modules. The image module loads the cover image. Readies the pixel data. The LSB engine puts in. Pulls out the payload. SQLite keeps logs and metadata and the local file system holds uploaded images and created stego images. This local secure-steganography system is built to keep data hidden in images.

1. Embedding Process (Data Hiding)

The embedding process safely hides an encrypted text Message inside a digital cover image using the Significant Bit (LSB) steganography technique.

Workflow:

1. Message and Key Input: You will enter a Message and provide an encryption Key.
2. Cover Image Upload: You will upload a supported image to be used as the cover image.
3. Message Encryption: The backend will encrypt the Message using the selected encryption algorithm.
4. Binary Conversion: The encrypted data will be converted into a binary bit sequence.
5. LSB Embedding: The LSB engine will embed the sequence into the least significant bits of the cover images pixel values.
6. Stego Image Generation: The modified image will be saved as a stego image. Provided to you for secure transmission.

2. Extraction Process (Data Recovery)

The extraction process retrieves the encrypted Message, from the stego image and decrypts it using the corresponding encryption key.

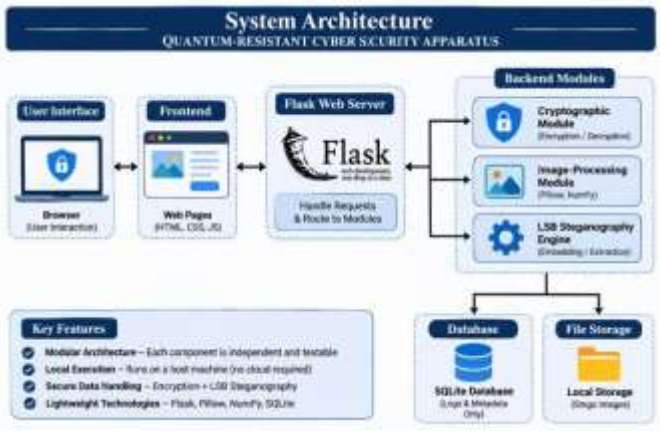
Workflow:

1. Stego Image Upload: The receiver will upload the image containing the hidden encrypted Message.
2. Pixel Data Processing: The image-processing module will read the pixel values of the image.
3. LSB Extraction: The LSB engine will extract the embedded sequence from the image pixels.
4. Payload Recovery: The extracted binary data will be reconstructed into its original encrypted byte format.
5. Decryption Key Input: The receiver will enter the matching encryption Key.
6. Message Decryption: The decryption module will decrypt the extracted payload.
7. Message Display: If extraction and decryption are successful the original secret Message will be displayed to the receiver.

VI. SYSTEM ARCHITECTURE

The proposed system architecture of the Quantum-resistant cyber security apparatus consists of several independent software modules that work together to provide secure message encryption and image-based data hiding using Least Significant Bit (LSB) steganography.

The architecture is designed as a locally hosted web application using Flask as the backend framework. The user interacts with the application through a web browser, which provides the interface for entering secret messages, uploading cover images, providing encryption keys, and retrieving hidden messages.



VII. CRYPTOGRAPHIC AND STEGANOGRAPHIC PROCESS

The report has a naming inconsistency. The architecture, testing and results sections keep referring to SM4.. The implementation section clearly says that the functions sm4_encrypt() and sm4_decrypt() actually use AES through the PyCryptodome library. They use ECB mode and a generated 16-byte key. Because of this this paper will temporarily refer to the method as AES-ECB. Before publication the actual source code must be checked to confirm the algorithm used the encryption mode, how the key is handled and whether the key is generated or entered by the user.

ECB mode has a weakness. It turns plaintext blocks into identical ciphertext blocks when using the same key. This does not offer the level of security that modern authenticated encryption modes provide. Using ECB should not be presented as a setup. A better version should use an authenticated encryption mode like AES-GCM.It should also use generated nonces and good key management practices.This change can only be claimed once it is actually implemented and tested.

LSB embedding hides data by changing the significant bits of pixels. The hidden message or payload must include its length. Have a clear way to know when it ends. Otherwise the extraction process won't know when to stop. The report does not give verified details, about

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.

the implementation. It does not explain the bit ordering, which channels are used how the payload is framed or how much data can be hidden. These details should come from the source code not be guessed at.

VIII. IMPLEMENTATION TECHNOLOGIES

The project report says Python is the programming language used. Flask is the backend web framework. For the frontend HTML, CSS and JavaScript are used. Pillow and NumPy help with image processing tasks. PyCryptodome is mentioned in the implementation part, for doing AES encryption and decryption. SQLite is used to store logs and metadata. Uploaded and generated files are saved on the file system. The application works on Windows and Linux. It can be accessed using any web browser.

IX. RESULTS AND DISCUSSION

The supplied report explains the outputs. The intended outputs include message encryption, embedding encrypted data into a cover image creating a stego image extracting data and decrypting the recovered payload. The supplied report also explains database logging and local file storage. These are system capabilities stated in the supplied report not independently verified findings.

The supplied report contains a sample test table. The sample test table lists results as "To be verified" while marking several cases as "Pass". These entries are not evidence of successful testing. Before publication each test should be carried out. The actual observed result written down. Useful measurements include embedding time, extraction time, input image dimensions, payload size, payload capacity, image quality after embedding and successful recovery rate across many test images. Any numerical result must be. Reproducible.

I see that the system has limitations. LSB data may be lost when images are compressed, resized, cropped or otherwise changed. Steganalysis may detect changes. The documented AES-ECB implementation requires review. Should not be described as authenticated encryption. Importantly the described combination does not provide formal post-quantum security.

X. CONCLUSION AND FUTURE WORK

This paper has described a web-based application that combines message encryption and LSB image steganography. The design uses Flask to coordinate the application, image-processing libraries to manipulate pixel data SQLite for metadata and local storage for image files. The approach demonstrates how encryption and data concealment can be combined in a workflow.

The project should not claim to be quantum-resistant on the basis of the implementation described in the report. Future work may include replacing the AES-ECB implementation, with authenticated encryption improving key management adding integrity-protected payload framing evaluating image quality and steganalysis resistance and integrating standardized post-quantum cryptographic mechanisms where appropriate. Each enhancement must be implemented, measured and documented before being reported as a result.

XI. REFERENCES

1) G. J. Simmons, “The prisoners’ problem and the subliminal channel,” in Advances in Cryptology: Proceedings of CRYPTO ’83, 1984. Verify edition and page details before final submission.
2) N. Provos and P. Honeyman, “Hide and seek: An introduction to steganography,” IEEE Security & Privacy, vol. 1, no. 3, pp. 32–44, May–June 2003.
3) National Institute of Standards and Technology, Advanced Encryption Standard (AES), FIPS PUB 197, 2001. Use the updated official edition applicable to the implementation.
4) PyCryptodome Documentation, “AES — Advanced Encryption Standard,” project documentation. Add the access date and exact documentation URL used.
5) Standardization Administration of China, “Information Security Technology—SM4 Block Cipher Algorithm,” GB/T 32907-2016, 2016.
6) M. H. K. Kombrink, Z. J. M. H. Geradts, and M. Worring, “Image Steganography Approaches and Their Detection Strategies: A Survey,” *ACM Computing Surveys.*
7) P. C. Mandal, I. Mukherjee, G. Paul, and B. N. Chatterji, “Digital Image Steganography: A Literature Survey,” *Information Sciences.*
8) J. Singh and M. Singla, “An Extensive Study of Digital Image Steganography Techniques,” *Indian Journal of Computer Science.*
9) “A Survey of Recent Advances in Image Steganography,” *Security and Privacy.*
10) National Institute of Standards and Technology, “Stateless Hash-Based Digital Signature Standard,” FIPS PUB 205, 2024.

© Author(s). This work is peer-reviewed, openly published, and permanently archived
This article is openly accessible and reusable with proper attribution.