Journal of Computing Research and Innovation (JCRINN) Vol. 8 No.2 (2023))
https://jcrinn.com : eISSN: 2600-8793 https://doi.org/10.24191/jcrinn.v8i2.373

Image Steganography Using Web Application

Nor Arzami Othman[1]*, Muhammad Nur Harith Shariffudin[2], Mohd Nizam Osman[3], Khairul Anwar Sedek[4]
[1,2,3,4] *College of Computing, Informatics, and Mathematics, Universiti Teknologi MARA Perlis Branch, Arau Campus, 02600 Arau, Perlis, Malaysia.*

*Corresponding author: *arzami@uitm.edu.my*

Received Date: 24 July 2023
Accepted Date: 28 July2023
Revised Date: 20 August 2023
Published Date: 1 September 2023

HIGHLIGHTS

- The web application project implementing image steganography holds considerable significance for various reasons such as it offers a practical and user-friendly platform that allows users to hide data within images seamlessly.
- Capability to enhance data security by making it challenging for unauthorized parties to detect or access hidden information. It facilitates tamper-proofing, as any unauthorized modifications to the image will likely corrupt the embedded data, thus alerting users to potential tampering attempts.
- By embedding data within images, the application optimizes the efficiency of image transmission. Users can now simultaneously transfer visual content and essential data, streamlining communication and reducing the need for separate data transfers.
- Empowers users by providing them with a convenient and accessible tool to enhance data protection, streamline communication, and safeguard the integrity of visual content. Its potential applications span various domains, from personal data security to professional communication and even sensitive information exchange in fields such as journalism, law enforcement, and cybersecurity.

ABSTRACT

Nowadays most people are workaholics, driven by intense peer pressure to break new ground. Therefore, rather than using books as their source material, photographic memories are kept as images. However, to protect sensitive information, reliable and secure communication techniques are necessary in many real-world situations. This research offers a user-friendly interface for securely embedding and extracting secret information within digital images. The primary goals are to design a steganographic algorithm for concealing data in images and to evaluate the usability of image steganography while keeping the quality of images and the access to decode the image. The application utilizes an advanced steganographic algorithm, which is Randomized Least Significant Bit (RLSB), to ensure robust data concealment while maintaining the visual integrity of the cover image. Users can upload their desired image, select the preferred steganographic algorithm, and encode the hidden data for added security. The web application supports encoding and decoding images for concealing data in images. To evaluate its performance, extensive testing was conducted, including embedding and extracting data using different image formats. The results demonstrated the application's effectiveness in hiding information while preserving image quality. The web application proved to be a versatile and practical tool with applications in various fields such as cryptography and digital forensics. In conclusion, the Image Steganography Web Application provides a convenient and secure solution for individuals and organizations needing to transmit sensitive data covertly within images, ensuring data privacy and integrity in an intuitive and user-friendly manner.

Keywords: *Steganographic, Randomized Least Significant Bit, Web Application, Hiding Information*

INTRODUCTION

In today's increasingly interconnected and digitized world, the need for secure communication and the protection of sensitive information has become paramount. Steganography, a fascinating field of study, provides a unique solution to these challenges. Among its various applications, image steganography stands out as a crucial technique for concealing information within digital images. It offers a powerful tool for covert communication and secure information exchange in a wide range of practical scenarios. One of the significant applications of image steganography is in the field of journalism. Journalists often find themselves in situations where they need to transmit sensitive information or evidence while safeguarding the anonymity of their sources. By employing steganographic techniques, journalists can embed hidden messages within images, making it difficult for unauthorized individuals to detect or intercept the information. This ensures that vital data remains confidential and protects the identities of those involved, thus preserving the integrity of journalistic work.

In the realm of cybersecurity, steganography serves as an additional layer of protection for sensitive data. Malicious actors constantly seek ways to intercept and exploit confidential information. By hiding encryption keys or confidential data within images using steganographic techniques, cybersecurity professionals can make it significantly more challenging for adversaries to detect and gain access to concealed information. This approach enhances the overall security of data transmission and reduces the risk of unauthorized access or data breaches. Law enforcement agencies also find image steganography invaluable for various purposes. They can embed watermarks or hidden identification markers within digital images, facilitating copyright protection or forensic investigations. By employing steganographic techniques, law enforcement professionals can discreetly embed unique markers or data within images, enabling them to trace the origin or ownership of digital content. This enhances their ability to protect intellectual property rights, prevent piracy, and conduct effective forensic analysis when necessary.

The widespread use of image steganography across diverse fields demonstrates its significance today. Whether it is for journalists ensuring the privacy of their sources, cybersecurity professionals safeguarding sensitive data, or law enforcement agencies protecting intellectual property, steganography plays a crucial role in secure communication. By leveraging the concealment capabilities of digital images, steganography empowers individuals and organizations to exchange information covertly and maintain the confidentiality and integrity of their data. As technology continues to advance, image steganography will remain an asset in the ongoing pursuit of secure and private communication.

PROBLEM STATEMENT

In numerous real-life scenarios, the demand for robust and secure communication methods becomes paramount to safeguard sensitive information. Conventional approaches like encryption may not always provide an adequate level of protection, as they can be vulnerable to interception or detection by malicious actors. The realm of information security encompasses a wide array of risks, ranging from software attacks and intellectual property infringements to the unauthorized disclosure of private data and information extortion (Li & Li, 2018). Steganography, as an alternative technique, offers a significant advantage by concealing a secret message within various media files (Agarwal & Malik, 2022). This covert embedding allows for a discreet transfer of information, making it more challenging for unauthorized individuals to

detect the existence of hidden data within the media files, thus providing an additional layer of security in communication.

OBJECTIVES

We are mainly focused on making a web application for concealing secret messages in photos. The project's primary goals are:

i. To design a steganographic algorithm for concealing data in images

ii. To evaluate the usability of image steganography while keeping the quality of images and the access to decode the image.

However, we are concerned about potential legal and ethical issues related to misuse and harm. There was also a risk of compromising users' privacy and security without their explicit consent (Rout & Mishra, 2014). Moreover, if the tool is misused, we could face legal liabilities as the developer. While the idea is intriguing, we must proceed with extreme caution and consider the potential risks and consequences before moving forward.

LITERATURE REVIEW

Steganography and Steganalysis

The history of steganography is extensive. The historical evidence shows that steganography may adapt to changing environmental conditions and advance nimbly. The analogue world, digital world, and Internet have all seen significant developments. The initial change was brought about by the creation of the computer, and it progressed to its current state with the growth of the Internet. The arrival of new media forms that might be exploited meant that each transition was inevitable. The many advantages network steganography has will encourage its use in the future (Seo, Manoharan & Mahanti, 2016).

Steganalysis serves as the countermeasure to steganography, aiming to identify the presence of steganographic content in digital devices and uncover any concealed messages. This process can be categorized into two main types: passive and active steganalysis. Digital forensics, a relatively recent field in Computer Science, concentrates on acquiring, preserving, and analysing digital evidence (Karampidis et al., 2018).

The LSB Techniques

The process of image steganography involves altering the least significant bit (LSB), which is the eighth bit inside an image pixel, to embed a secret message. In the case of a 24-bit image, each pixel comprises three color components: red, green, and blue, each represented by a byte. By modifying one bit in each color component, it becomes possible to store 3 bits of hidden data in a single pixel.

As there are 256 possible intensities for each primary color, altering the least significant bit (LSB) of a pixel results in minor changes in color intensity that are imperceptible to the human eye. This successful concealment allows the message to remain hidden even when utilizing the LSB of a well-chosen image without any noticeable difference (Lokeswara Reddy et al., 2011).

METHODOLOGY

The Waterfall Model, being the oldest and most renowned Software Development Life Cycle (SDLC) model, finds extensive application in government projects and large corporations. Notably, this model follows a sequential approach, progressing step-by-step through the phases of requirements analysis, design, coding, testing, and maintenance (Alshamrani & Bahattab, 2015).

The Waterfall Model of SDLC was effectively utilized in the development of the Image Steganography Web Application, offering a structured and systematic approach to the project. Given the sequential nature of the Waterfall Model, it aligned well with the distinct phases involved in building the application. The initial phase involved requirements analysis, where thorough gathering and analysis of project requirements took place. This included identifying the specific steganography techniques to be implemented, supported image file formats, user interface requirements, and any security considerations related to data encryption and user authentication.

The design phase focused on conceptualizing the application's architecture and user interface. This stage involved creating a detailed plan for the steganography algorithms' integration, selecting appropriate technologies, and laying out the user interface for seamless interaction and user experience. The design phase aimed to create a blueprint that guided the development process effectively. Once the design was finalized, the coding phase commenced, where we started implementing the steganography algorithms, user interface components, and other functionalities. The application's backend and front end were constructed according to the design specifications. The sequential nature of the Waterfall Model ensured that each development phase built upon the previous one, creating a solid foundation for subsequent stages.

After completing the coding phase, the application underwent rigorous testing to verify its functionality and ensure that hidden messages could be encoded and extracted accurately. Testing scenarios encompassed various image formats, data sizes, and edge cases to validate the application's reliability and accuracy. Any issues or bugs identified during testing were addressed before moving to the next phase. Following successful testing and deployment, the application entered the maintenance phase, where updates, bug fixes, and improvements were continuously made based on user feedback and changing requirements. This stage ensured the application remained robust, secure, and relevant over time. The linear and structured approach of the Waterfall Model in the development of the Image Steganography Web Application ensured a clear progression through each stage, enabling better project planning, resource management, and risk mitigation.

The cover media contains an embedded secret message that must be transmitted as illustrated in Figure 1. The stego key is then employed to increase security. The cover media contains a hidden message that has been embedded using any steganography algorithm. The resulting file is known as stego media, and it is sent to the receiving side. The secret message is extracted from the stego media at the receiver side using the stego key.

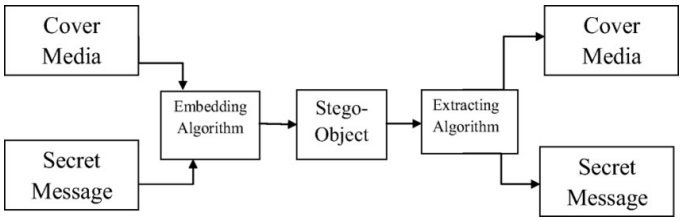

Figure 1: Block diagram of steganography

In steganography, the hidden data is typically encrypted before being embedded into the cover data, adding an extra layer of security as shown in Figure 2. This encryption ensures that even if an attacker discovers the presence of hidden data, they cannot access it without the decryption key. Additionally, encryption disguises the hidden data as random noise, preventing detection by the attacker. This process, known as steganography encryption, ensures confidentiality, safeguarding the privacy of the hidden data. To access the secret data, the recipient must use the decryption key to decrypt the embedded data and then employ steganography techniques to extract the secret data from the cover data.

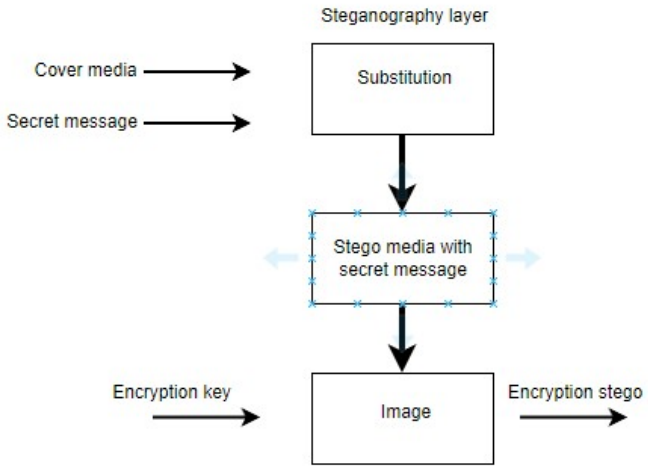

Figure 2: Embedding/Encryption Process

ALGORITHM DESIGN

In designing the algorithm for the Image Steganography Web Application, the focus is on creating efficient and secure methods for embedding and extracting hidden messages within digital images. The process involves selecting a suitable encoding algorithm like Least Significant Bit (LSB) or Random LSB (RLSB), determining the embedding capacity based on the image's color depth and resolution, and converting the secret message to binary format for embedding. The algorithm ensures minimal visual distortion while modifying the least significant bits (LSBs) of the image pixels to represent the secret message. For extraction, the corresponding decoding algorithm is used to retrieve the hidden message by extracting the modified LSBs. By integrating this algorithm seamlessly into the web application, users can securely and discreetly communicate through steganographic images. Figure 3 and Figure 4 show the coding snippets of the encoding algorithm in Python language that implements LSB and RLSB.

```python
def encode_data_lsb(image, data):
    binary_data = ''.join(format(ord(char), '08b') for char in data)
    pixels = list(image.getdata())
    num_channels = len(pixels[0])

    encoded_pixels = []
    data_index = 0
    for pixel in pixels:
        if data_index >= len(binary_data):
            encoded_pixels.append(pixel)
            continue

        modified_pixel = list(pixel)
        for channel in range(num_channels):
            if data_index < len(binary_data):
                modified_pixel[channel] = (modified_pixel[channel] & 0xFE) | int(binary_data[data_index])
                data_index += 1

        encoded_pixels.append(tuple(modified_pixel))

    encoded_image = Image.new(image.mode, image.size)
    encoded_image.putdata(encoded_pixels)

    return encoded_image
```

Figure 3: LSB Encode Algorithm

```python
def encode_data_rlsb(image, data):
    binary_data = ''.join(format(ord(char), '08b') for char in data)
    pixels = list(image.getdata())
    num_channels = len(pixels[0])

    encoded_pixels = []
    data_index = 0
    for pixel in pixels:
        if data_index >= len(binary_data):
            encoded_pixels.append(pixel)
            continue

        modified_pixel = list(pixel)
        for channel in range(num_channels):
            if data_index < len(binary_data):
                bit_to_hide = int(binary_data[data_index])
                if random.random() < 0.5:
                    modified_pixel[channel] = (modified_pixel[channel] & 0xFE) | bit_to_hide
                else:
                    modified_pixel[channel] = (modified_pixel[channel] & 0xFD) | (bit_to_hide << 1)
                data_index += 1

        encoded_pixels.append(tuple(modified_pixel))

    encoded_image = Image.new(image.mode, image.size)
    encoded_image.putdata(encoded_pixels)

    return encoded_image
```

Figure 4: RLSB Encode Algorithm

The coding snippets of the decoding algorithm in Python language, implementing LSB and RLSB, are illustrated in Figure 5 and Figure 6 respectively.

```python
def decode_data_lsb(image):
    pixels = list(image.getdata())
    num_channels = len(pixels[0])

    binary_data = ""
    for pixel in pixels:
        for channel in range(num_channels):
            binary_data += str(pixel[channel] & 1)

    decoded_data = ""
    for i in range(0, len(binary_data), 8):
        byte = binary_data[i:i + 8]
        decoded_data += chr(int(byte, 2))

    return decoded_data
```

Figure 5: LSB Encode Algorithm

```python
def decode_data_rlsb(image):
    pixels = list(image.getdata())
    num_channels = len(pixels[0])

    binary_data = ""
    for pixel in pixels:
        for channel in range(num_channels):
            bit_to_retrieve = pixel[channel] & 1
            binary_data += str(bit_to_retrieve)

    decoded_data = ""
    for i in range(0, len(binary_data), 8):
        byte = binary_data[i:i + 8]
        decoded_data += chr(int(byte, 2))

    return decoded_data
```

Figure 6: RLSB Encode Algorithm

Python was used because it is widely favored for steganography due to its versatility as a programming language and the abundance of processing image libraries like the Pillow library. With such libraries readily available, handling image data and implementing steganography algorithms becomes more convenient (Goel, 2020). Alongside the implementation of classic ciphers and secure communication protocols like SSL/TLS, Python serves as a versatile tool for securing sensitive data and communications (Bowne, 2018).

Web applications are software programs that users can access through web browsers over a network. It was created using browser-supported languages like HTML and JavaScript. These applications rely on web browsers for execution and offer various functionalities, including online retail sales, auctions, and webmail services (Al-Fedaghi, 2011).

Flask is a popular and lightweight web development framework for Python (Rama Vyshnavi & Malik, 2019). It offers a simple and elegant approach to building web applications, making it an excellent choice for developers seeking to create applications quickly and efficiently. With its minimalistic design, Flask provides essential features, leaving developers the flexibility to choose and integrate other tools as needed. Its modular nature allows for easy customization and scalability (Grinberg, 2018).

FINDINGS AND DISCUSSIONS

The final acceptance test for the Image Steganography Web Application holds utmost significance as it serves as a critical milestone in the development process, ensuring that the application fulfills all the specified requirements and operates as intended. This comprehensive testing phase validates the application's performance, functionality, and usability, thereby confirming its readiness for deployment and use by end-users. By conducting rigorous testing scenarios and real-world simulations, the acceptance test assesses the application's ability to handle various inputs, process image steganography operations efficiently, and safeguard data integrity. It also verifies that the user interface is intuitive, visually appealing, and user-friendly, ensuring a seamless and enjoyable experience for users. Ultimately, the successful completion of the final acceptance test gives the green light to proceed with the deployment of the Image Steganography Web Application, instilling confidence in its reliability and functionality for secure communication and data protection.

Functional Testing verifies that all the core functionalities of the web application work correctly. It includes testing the image upload and processing, message embedding, decoding, and the selection and application of encoding algorithms (e.g., LSB, RLSB). Each feature is thoroughly tested to ensure it functions as expected.

Compatibility Testing was performed across different web browsers is a crucial step in ensuring its compatibility and seamless functionality across various platforms. Three widely used web browsers, namely Chrome, Firefox, and Edge, were selected for thorough testing to verify that the application not only works as intended but also maintains a consistent and visually appealing appearance on different browsers. During the testing process, the application was carefully assessed for any potential issues or discrepancies that might arise due to browser-specific variations. This involved conducting a series of test cases and scenarios on each browser to evaluate its performance in handling image steganography functionalities. The tests covered the core features of the application, such as image embedding and extraction, encryption and decryption, user authentication, and user interface interactions. The results of the testing revealed that both Chrome and Firefox successfully handled the Image Steganography Web

Application with consistent performance and accuracy. The application demonstrated seamless functionality, regardless of the browser used.

Likewise, Microsoft Edge also proved to be compatible with the application, delivering a reliable user experience and smooth operations. By ensuring compatibility across these popular web browsers, the Image Steganography Web Application becomes accessible to a broader audience of users. Whether they prefer Chrome, Firefox, or Edge, users can confidently utilize the application to securely hide and extract messages within digital images. This cross-browser compatibility guarantees a consistent and reliable experience, enhancing user satisfaction and encouraging wider adoption of the steganography tool. As a result, the application can effectively serve its intended purpose of enabling secure and confidential communication through steganographic techniques.

Overall, the comprehensive testing across Chrome, Firefox, and Edge validated the Image Steganography Web Application's versatility and robustness. The successful performance of these major web browsers reassured users that they could utilize the application on their preferred platforms without any hindrance. This broad cross-browser compatibility further enhanced the application's accessibility and usability, making it an effective and user-friendly tool for secure communication through image steganography. Additionally, the test assesses the compatibility of various image file formats, including JPG, PNG, GIF, and WEBP. Table 1 presents the results of this compatibility evaluation.

Table 1: Results of file format test

File Format	**Result**
JPG	*Pass*
PNG	*Pass*
GIF	*Fail*
WEB	*Pass*

GIF (Graphics Interchange Format) fails as an image file format for image steganography due to its limited color depth and lossless compression. GIF images support only 256 colors, which results in a lower capacity for hiding large amounts of data compared to other formats like JPG or PNG. Additionally, GIF uses a lossless compression technique, which means that the image data remains unchanged during compression and decompression. As a result, any modifications made to the GIF image, such as embedding a hidden message, may cause noticeable visual artifacts, making the steganography less effective in preserving the original image's quality and hiding the secret data discreetly. For these reasons, GIF is not an ideal choice for image steganography applications that require higher data capacity and minimal visual distortion.

Performance Testing for file size is a crucial aspect of assessing the efficiency and scalability of the Image Steganography Web Application. This testing involves subjecting the application to various file sizes, ranging from small to large, and measuring the time it takes to encode and decode the images with hidden messages. During the performance testing, it was evident that larger file sizes indeed resulted in longer decoding times. This observation can be attributed to the inherent complexities introduced by the size of the data being processed. As the size of the encoded image increases, the steganography algorithm has to handle a larger amount of data, requiring more computational resources and time for the extraction of the hidden message. The process involves traversing through an increased number of pixels, manipulating color values, and analyzing a larger number of bits to extract the embedded information accurately.

Additionally, larger files often have higher color depths and resolutions, which further impact the decoding process. Higher color depths mean that each pixel contains more bits, making the extraction of the hidden

data more intricate. Similarly, higher resolutions imply that the image has a greater number of pixels, resulting in an exponential increase in the volume of data that needs to be processed during decoding. To mitigate the potential performance issues associated with larger file sizes, it is required to optimize their algorithms for efficiency. Balancing the trade-off between the amount of data that can be hidden within an image and the decoding time is essential. Algorithms can be fine-tuned to reduce the computational burden without significantly compromising the capacity for hidden data.

Moreover, parallel processing techniques and hardware acceleration can be explored to speed up the decoding process. By leveraging the power of modern multi-core processors or specialized hardware like GPUs, the application can efficiently process large image files with minimal decoding time. In conclusion, the performance testing for file size in the Image Steganography Web Application highlighted the need for thoughtful algorithm design and optimization to handle larger files efficiently. As image steganography continues to evolve, researchers and developers must continue to address these challenges to ensure the seamless integration of secure and efficient data hiding within digital images, regardless of the file size.

CONCLUSION AND RECOMMENDATIONS

In conclusion, the development of the Image Steganography Web Application has been a significant endeavor, providing a powerful tool for secure and covert communication through digital images. Throughout the project, various steganography techniques were explored, and the application was designed to ensure user-friendly functionality while maintaining data integrity and security. With continuous improvements and advancements in the field of steganography, the application can serve as an asset in safeguarding sensitive information and fostering privacy in the ever-evolving digital landscape.

In the future development of the Image Steganography Web Application, key areas of focus include enhancing the user interface based on user feedback and visual representations. More advanced steganographic techniques can be applied to enhance the capacity and security of embedding hidden messages within digital images. By exploring cutting-edge developments in steganography, it can be enriched with sophisticated algorithms that offer greater data-hiding capabilities and improved resistance against potential attacks.

Finally, the integration of steganography capabilities within popular social media platforms can revolutionize the way users communicate securely and discreetly. By seamlessly incorporating steganographic features into these platforms, users would gain the ability to encode hidden messages within images and share them with select recipients through familiar and widely used interfaces. This integration has the potential to enhance privacy and confidentiality, as steganographic images blend in seamlessly with regular image content, avoiding suspicion or detection. Embracing this advancement could foster a new era of secure and covert communication, empowering users with increased control over their digital interactions while maintaining the convenience and familiarity of mainstream social media platforms.

ACKNOWLEDGMENTS

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

CONFLICT OF INTEREST DISCLOSURE

The authors declared that they have no conflicts of interest to disclose.

REFERENCES

Agarwal, A., & Malik, S. (2022). A Brief Review on Various Aspects of Steganography Followed by Cryptographic Analysis. *2022 IEEE 7th International Conference for Convergence in Technology, I2CT 2022*. https://doi.org/10.1109/I2CT54291.2022.9825422

Al-Fedaghi, S. (2011). Developing Web Applications. *International Journal of Software Engineering and Its Applications*, *5*(2), 57–68.

Alshamrani, A., & Bahattab, A. (2015). A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. *IJCSI International Journal of Computer Science Issues*, *12*(1), 106–111. www.IJCSI.org

Bowne, S. (2018). Hands-On Cryptography with Python. Packt Publishing.

Goel, A. (2020, August 20). Image-based Steganography using Python. Geeksforgeeks. https://www.geeksforgeeks.org/image-based-steganography-using-python/

Grinberg, M. (2018). Flask Web Development: Developing Web Applications With Python (2nd ed.). O'Reilly Media.

Karampidis, K., Kavallieratou, E., & Papadourakis, G. (2018). A review of image steganalysis techniques for digital forensics. *Journal of Information Security and Applications*, *40*, 217–235. https://doi.org/10.1016/j.jisa.2018.04.005

Li, X., & Li, H. (2018). A Visual Analysis of Research on Information Security Risk by Using CiteSpace. *IEEE Access*, *6*, 63243–63257. https://doi.org/10.1109/ACCESS.2018.2873696

Lokeswara Reddy, V., Subramanyam, A., & Reddy, P. C. (2011). Implementation of LSB Steganography and its Evaluation for Various File Formats. *Int. J. Advanced Networking and Applications*, *2*(5), 868–872.

Rama Vyshnavi, V., & Malik, A. (2019). Efficient Way of Web Development Using Python and Flask. *International Journal of Recent Research Aspects*, *6*(2), 16–19.

Rout, H., & Mishra, B. K. (2014). Pros and Cons of Cryptography, Steganography and Perturbation Techniques. *IOSR Journal of Electronics and Communication Engineering*, 2278–8735. www.iosrjournals.org

Seo, J. O., Manoharan, S., & Mahanti, A. (2016). Network steganography and steganalysis - a concise review. *2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology (ICATccT)*, 368–371.