National College of Ireland

Configuration Manual

MSc Research Project
MSc Cybersecurity TopUp (MSCCYBETOP)

Rajni Priya
Student ID: x24241946

School of Computing
National College of Ireland

Supervisor: Mark Monaghan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Rajni Priya
Student ID:	x24241946
Programme:	MSc Cybersecurity TopUp (MSCCYBETOP) **Year:** 2024-25
Module:	MSc (Research) Practicum
Supervisor:	Mark Monaghan
Submission Due Date:	31-07-2025
Project Title:	A Secure Web Application for Data Storage and Retrieval Using Multi-Level Protection
Word Count:	411 **Page Count:** 6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Rajni Priya

Date: 31-07-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Rajni Priya
x24241946

1 Introduction

A system that stores images after encrypting it by adding by using AES algorithm, RSA algorithm, steganography and noise addition. The images contains information which is secured. The secured information is stored(uploaded to the system) and retrieved when required. This manual contains the hardware and software specifications for carrying out the study defined, in section 2. Section 3 contains the details of the technology and software used, here. Section 4 contains the details of the implementation of the secure storage and retrieval of the encrypted data.

2 Experimental Setup

For carrying out this experiment a high-end PC was needed and a PC that met the requirements for carrying out this study.

- **Hardware specifications:** The PC used in this study contained an Intel I5 third gen processor with 8 GB RAM, including a 224 GB SSD.
- **Operating System:** Windows 10
- **Experimental Setup:** For running the artifact built here, Windows 10, Python 3.6, Anaconda 3 and VS code.

3 Technologies and software used for implementation

Software

Anaconda is an open-source distribution for Python that makes package management easier In Anaconda the package versions are managed using conda package management system, this was done to ensure that existing frameworks and packages are not disrupted by analyzing the current environment (Rolon-Mérette *et al.*, 2020). Anaconda provides extra open-source packages through PyPI and the conda package and virtual environment manager.

Python is an open-source and free programming language. Python was used because it provides libraries that can be used for the implementation of different kinds of encryption and decryption algorithms. Python has a large and active community support which can help in solving problems related to any libraries in Python (Tahmooresi, Heydarnoori and Aghamohammadi, 2020). So, any issues that were encountered can be solved.

Python Django was the web framework used in the study because it helped in the fast building of a web app, the use of this framework meant that the web app was built quickly in this study (Kumar and Nandal, 2024).

4 Implementation

Uploading

Step 1: Importing libraries for encryption

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP, AES
from Crypto.Util.Padding import pad, unpad
import base64
import cv2
import numpy as np
from PIL import Image
import io
import os
import struct
```

Step 2**:** Key generation for RSA

```
[ ]  def generate_rsa_keys():
         """Generate RSA public and private keys"""
         key = RSA.generate(2048)
         private_key = key.export_key()
         public_key = key.publickey().export_key()
         return private_key, public_key
```

Step 3: RSA data encryption

```
[ ]  def encrypt_data(data, public_key):
         """Encrypt data using RSA-OAEP"""
         rsa_key = RSA.import_key(public_key)
         cipher = PKCS1_OAEP.new(rsa_key)
         encrypted_data = cipher.encrypt(data.encode('utf-8'))
         return base64.b64encode(encrypted_data).decode('utf-8')
```

Step 4: Compressed data

```
[ ]  def compress_data(data):
         """Compress data into a PNG image"""
         if isinstance(data, str):
             data_bytes = data.encode('utf-8')
         else:
             data_bytes = data

         array_length = len(data_bytes)
         width = int(np.ceil(np.sqrt(array_length)))
         height = width

         padded_data = np.pad(np.frombuffer(data_bytes, dtype=np.uint8),
                              (0, width * height - array_length),
                              mode='constant')

         img = padded_data.reshape((height, width))
         img_pil = Image.fromarray(img, mode='L')

         buffer = io.BytesIO()
         img_pil.save(buffer, format="PNG")
         return buffer.getvalue()
```

Step 5: Embedding data into image

```
def embed_data_into_image(data, mask_image_path, output_image_path):
    """Hide data in an image using LSB steganography"""
    mask_image = cv2.imread(mask_image_path)
    if mask_image is None:
        raise ValueError("Could not load mask image")

    data_bytes = np.frombuffer(data, dtype=np.uint8)
    data_length = len(data_bytes)

    if data_length + 4 > mask_image.size:
        raise ValueError("Data too large for the mask image")

    flat_image = mask_image.flatten()

    # Store data length in first 4 bytes
    length_bytes = data_length.to_bytes(4, byteorder='big')
    flat_image[:4] = np.frombuffer(length_bytes, dtype=np.uint8)

    # Store the actual data
    flat_image[4:4+data_length] = data_bytes

    stego_image = flat_image.reshape(mask_image.shape)
    cv2.imwrite(output_image_path, stego_image)
    return stego_image
```

Step 6: AES encryption

```
def aes_encrypt(data, key):
    """Encrypt data using AES in CFB mode"""
    iv = os.urandom(16)
    cipher = AES.new(key, AES.MODE_CFB, iv=iv)
    encrypted_data = cipher.encrypt(pad(data, AES.block_size))
    return iv + encrypted_data
```

Step 7: Adding noise

```
def add_laplace_noise(data, mean=0, scale=0.25):
    """Add Laplace noise to data for additional security"""
    noise = np.random.laplace(mean, scale, data.shape).astype(np.int16)
    noisy_data = np.clip(data + noise, 0, 255).astype(np.uint8)
    return noisy_data, noise
```

Step 8: Loading an image

```
def load_image(image_path):
    """Load an image into a numpy array"""
    image = Image.open(image_path).convert('RGB')
    return np.array(image)
```

Step 9: Saving encrypted image

```
def save_shape(shape, file_path):
    """Save image dimensions to a file"""
    with open(file_path, "wb") as f:
        f.write(struct.pack('III', *shape))
```

Retrieval

Step 1: Loading encrypted image.

```
def load_image(image_path):
    """Load an image into a numpy array"""
    image = Image.open(image_path).convert('RGB')
    return np.array(image)
```

Step 2: AES decryption

```
def aes_decrypt(data, key):
    """Decrypt data using AES in CFB mode"""
    iv = data[:16]
    encrypted_data = data[16:]
    cipher = AES.new(key, AES.MODE_CFB, iv=iv)
    return unpad(cipher.decrypt(encrypted_data), AES.block_size)
```

Step 3: Noise removal

```
def remove_laplace_noise(noisy_data, noise):
    """Remove Laplace noise from data"""
    noisy_data = noisy_data.astype(np.int16)
    noise = noise.astype(np.int16)
    clean_data = np.clip(noisy_data - noise, 0, 255).astype(np.uint8)
    return clean_data
```

Step 4: Extract data from image

```
def extract_data_from_image(stego_image_path):
    """Extract hidden data from an image"""
    stego_image = cv2.imread(stego_image_path)
    if stego_image is None:
        raise ValueError("Could not load stego image")

    flat_image = stego_image.flatten()

    # Read data length from first 4 bytes
    length_bytes = flat_image[:4].tobytes()
    data_length = int.from_bytes(length_bytes, byteorder='big')

    # Read the actual data
    data_bytes = flat_image[4:4+data_length].tobytes()
    return data_bytes
```

Step 5: Decompressing data

```
def decompress_data(compressed_data):
    """Decompress data from a PNG image"""
    img_pil = Image.open(io.BytesIO(compressed_data))
    img_array = np.array(img_pil)
    data_bytes = img_array.flatten().tobytes()
    return data_bytes.rstrip(b'\x00')
```

Step 6: Decrypting data

```
def decrypt_data(encrypted_data, private_key):
    """Decrypt data using RSA-OAEP"""
    rsa_key = RSA.import_key(private_key)
    cipher = PKCS1_OAEP.new(rsa_key)
    encrypted_bytes = base64.b64decode(encrypted_data)
    decrypted_data = cipher.decrypt(encrypted_bytes)
    return decrypted_data.decode('utf-8')
```

References

Rolon-Mérette, D. *et al.* (2020) 'Introduction to Anaconda and Python: Installation and setup,' *The Quantitative Methods for Psychology*, 16(5), pp. S3–S11. https://doi.org/10.20982/tqmp.16.5.s003.

Tahmooresi, H., Heydarnoori, A. and Aghamohammadi, A. (2020) 'An analysis of Python's topics, trends, and technologies through mining stack Overflow discussions,' *arXiv (Cornell University)* [Preprint]. https://doi.org/10.48550/arxiv.2004.06280.

Kumar, N.M. and Nandal, N.D.R. (2024) 'Python's Role in Accelerating Web Application Development with Django,' *Deleted Journal*, 2(06), pp. 2092–2105. https://doi.org/10.47392/irjaem.2024.0307.