

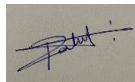
**National College of Ireland  
Project Submission Sheet**

**Student Name:** Rahul Sureshababu Pillai  
**Student ID:** X23103892  
**Programme:** MSc Cybersecurity **Year:** 2024  
**Module:** Msc Research Project  
**Lecturer:** Mark Monagan  
**Submission Due Date:** 12/08/2024  
**Project Title:** Configuration Manual  
**Word Count:** 385

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 references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

**Signature:**



**Date:** 12/08/2024

**PLEASE READ THE FOLLOWING INSTRUCTIONS:**

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# AI Acknowledgement Supplement

[Msc Research Project Configuration Manual]

Your Name/Student Number	Course	Date
Rahul Pillai	MSc Cybersecurity	12/08/2024

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click [here](#).

## AI ACKNOWLEDGMENT

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

Tool Name	Brief Description	Link to tool
NAH		

## DESCRIPTION OF AI USAGE

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used.**

[Insert Tool Name]	
[N.A]	
[Insert Sample prompt]	[Insert Sample response]

## EVIDENCE OF AI USAGE

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.

### ADDITIONAL EVIDENCE:

[N.A]

### ADDITIONAL EVIDENCE:

[N.A]

# Table of Contents

1	Introduction.....	4
2	Experimental Setup.....	4
3	Technologies and Software used for Implementation.....	4
4	Implementation.....	4
	Reference.....	7

## 1 INTRODUCTION

This configuration manual explains everything about the tools and technologies used in this project. Section 2 gives an overview of how the experiments were set up. Section 3 describes the different technologies and software tools used. Section 4 provides a step-by-step guide for ECC encryption, Huffman compression, LSB steganography, storing data with IPFS, and the reverse processes for each. Lastly, Section 5 lists the references used for this software guide.

## 2 EXPERIMENTAL SETUP

The experiment was carried out on a personal computer that was set up specifically for this project. This setup was chosen to make sure everything ran smoothly and performed well for all the tasks needed.

- **Hardware Specifications:** Intel i5 processor 7<sup>th</sup> generation, 8GB RAM, 256GB SSD.
- **Operating System:** Windows 10
- **Experimental Setup:** Anaconda 2.4.2 with Python 3.6

## 3 TECHNOLOGIES AND SOFTWARE USED FOR IMPLEMENTATION

- **Software Used:** Anaconda 2.4.2 with Python 3.6
- **Anaconda:** Anaconda is a versatile software platform known for its support of Python and R programming languages, including cryptography. It offers robust package and environment management capabilities, which simplify the setup and maintenance of diverse development environments. Anaconda is highly favored among data scientists, researchers, and developers for its extensive library of pre-installed tools and packages essential for various computational tasks, including data analysis, machine learning model development, scientific computing, and cryptographic applications (Root, 2021).

## 4 IMPLEMENTATION

Step 1: Anaconda was downloaded and installed

Step 2: Import necessary libraries to execute information protection

Step 3: Encrypt the text using ECC encryption algorithm.

```
def encrypt_ECC(msg, pubKey):
    ciphertextPrivKey = secrets.randbelow(curve.field.n)
    sharedECKKey = ciphertextPrivKey * pubKey
    secretKey = ecc_point_to_256_bit_key(sharedECKKey)
    ciphertext, nonce, authTag = encrypt_AES_GCM(msg, secretKey)
    ciphertextPubKey = ciphertextPrivKey * curve.g
    return (ciphertext, nonce, authTag, ciphertextPubKey)
```

**Figure (1): Encrypt text**

Step 4: Compress the encrypted text with the help of Huffman encoding algorithm.

```
def compress_ciphertext(ciphertext):
    binary_ciphertext = ''.join(format(byte, '08b') for byte in ciphertext)

    frequency_table = build_frequency_table(binary_ciphertext)

    huffman_tree = build_huffman_tree(frequency_table)

    huffman_codes = {}
    build_huffman_codes(huffman_tree, '', huffman_codes)

    compressed_ciphertext = ''.join(huffman_codes[symbol] for symbol in binary_ciphertext)

    return compressed_ciphertext, huffman_codes, huffman_tree
```

**Figure (2) Compress Encrypted text**

Step 5: After compressing, hide the text message into a image. This is achieved by applying LSB encoding technique.

```
def hide_text_in_image(filename):
    with open('outputs/compressed_ciphertext.txt', 'rb') as f:
        compressed_ciphertext = f.read()
    image_path = filename
    image = Image.open(image_path)
    width, height = image.size

    binary_ciphertext = ''.join(format(byte, '08b') for byte in compressed_ciphertext)
    binary_ciphertext += '0' * ((width * height * 3) - len(binary_ciphertext))

    index = 0
    for y in range(height):
        for x in range(width):
            pixel = list(image.getpixel((x, y)))

            for i in range(3):
                if index < len(binary_ciphertext):
                    pixel[i] = (pixel[i] & 0xFE) | int(binary_ciphertext[index])
                    index += 1

            image.putpixel((x, y), tuple(pixel))

    stego_image_path = 'outputs/stego_image.png'
    image.save(stego_image_path)
```

**Figure (3) Steg Compressed text into image**

Step 6: The steganographic image is then stored at IPFS(Inter Planetary File System).

```
def enc(e1):
    global filename, hash1
    text = e1.get("1.0", "end-1c")
    encrypt_text(text)
    print("Text encrypted successfully.")

    with open('outputs/enc_text.txt', 'rb') as f:
        ciphertext = f.read()
    compress_and_save_ciphertext(ciphertext)
    print("Ciphertext compressed and saved successfully.")
    hide_text_in_image(filename)
    print("Steganography process completed")
    api = ipfshttpclient.connect('/ip4/127.0.0.1/tcp/5001/http')
    new_file = api.add('outputs/stego_image.png')
    hash1 = new_file.get('Hash')
    messagebox.showinfo("", "Finished")
```

**Figure (4) Store Stego Image in IPFS**

Step 7: The stored image is retrieved back from the IPFS.

Step 8: By applying LSB decoding to the fetched image, desteg it and obtain the compressed image.

```
def retrieve_text_from_image(stego_image_path):

    stego_image = Image.open(stego_image_path)
    width, height = stego_image.size

    binary_ciphertext = ''

    for y in range(height):
        for x in range(width):
            pixel = stego_image.getpixel((x, y))

            for i in range(3):
                binary_ciphertext += str(pixel[i] & 0x01)

    binary_ciphertext = binary_ciphertext.rstrip('0')
    ciphertext = bytearray()

    for i in range(0, len(binary_ciphertext), 8):
        byte = binary_ciphertext[i:i + 8]
        ciphertext.append(int(byte, 2))

    with open('outputs/retrieved_compressed.txt', 'wb') as f:
        f.write(ciphertext)
```

**Figure (5) Get text from image**

Step 9: The text message is decompressed using Huffman decompression algorithm.

```
def decompress_ciphertext(compressed_ciphertext, huffman_codes, huffman_tree):
    binary_ciphertext = ''
    current_code = ''
    for bit in compressed_ciphertext:
        current_code += bit
        if current_code in huffman_codes:
            symbol = huffman_codes[current_code]
            binary_ciphertext += symbol
            current_code = ''

    ciphertext = bytearray()
    for i in range(0, len(binary_ciphertext), 8):
        byte = binary_ciphertext[i:i + 8]
        ciphertext.append(int(byte, 2))

    return ciphertext
```

**Figure (6) Decompress text**

Step 10: Finally, the decompressed text decrypt and sees the message.

```
def decrypt_ECC(encryptedMsg, privKey):  
    (ciphertext, nonce, authTag, ciphertextPubKey) = encryptedMsg  
  
    sharedECCKey = privKey * ciphertextPubKey  
    secretKey = ecc_point_to_256_bit_key(sharedECCKey)  
    plaintext = decrypt_AES_GCM(ciphertext, nonce, authTag, secretKey)  
    return plaintext
```

**Figure (7) Decrypt Text**

## REFERENCE

Root, D. (2021) 'An overview of the Anaconda distribution - towards data science,' Medium, 16 December.  
<https://towardsdatascience.com/an-overview-of-the-anaconda-distribution-9479ff1859e6>.