Configuration Manual

MSc Research Project
Cyber Security

Shubham Karodimal Parakh
Student ID: X21154376

School of Computing
National College of Ireland

Supervisor: Dr. Rohit Verma

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Shubham Karodimal Parakh		
Student ID:	X21154376		
Programme:	MSc in Cyber Security	**Year:**	2022-2023
Module:	MSc Internship Project		
Lecturer:	Rohit Verma		
Submission Due Date:	15/12/2022		
Project Title:	Securing passwords storage using image steganography by implementing AES encryption and Argon2 hashing		
Word Count:	**470**	**Page Count: 7**	

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: ………

Date: 14/12/2022

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

Shubham Karodimal Parakh
X21154376

1 Introduction

This handbook includes information on the setup and prerequisites for the suggested model, including specifics libraries and required software. The setup manual also provides instructions on how to use the algorithms required to create the suggested model.

2 System Configurations

The suggested model employs a number of methods, including LSB image steganography, argon2 hashing, and AES-256 encryption. These methods make use of power efficient Python packages and layers. We need a system with strong configurations in order to swiftly execute the complete procedure. Hence, we utilized a very powerful laptop to match the needs of the model for the same reason.

2.1 Device Specification

2.2 Device 1:

Processor	Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
Installed RAM	16.0 GB (15.9 GB usable)
System Type	64-bit operating system, x64-based processor
SSD	446 GB SSD
Graphics card	NVIDIA GeForce RTX 3050

2.3 Device 2:

Processor	Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
Installed RAM	8.0 GB (7.9 GB usable)
System Type	64-bit operating system, x64-based processor
SSD	512 GB SSD
Graphics card	NVIDIA GeForce MX130

2.4 Software and tools

Operating System: Windows 11 Home Single Language
Python: Python 3.10.9
Visual Studio Code
numpy version: 1.23.4

3 Implementation

This start by importing libraries that are required to run the proposed model which uses different techniques like hashing, encryption and steganography

```
main.py > ...
import csv
import sys
import time
from random import *
from pprint import pprint
import aes
from os import urandom
from ar import Arg_Hash
import stegno
import timeit
import avalancheEffect
import base64

import hashlib
from Crypto import Random
from Crypto.Cipher import AES

from argon2 import PasswordHasher

import cv2
import numpy as np
from PIL import Image
```

Figure-1 Libraries

Once the libraries are imported we start by performing argon2 hashing on the lists of passwords as shown in figure-2

```
from argon2 import PasswordHasher

class Arg_Hash:
    def __init__(self) -> None:
        self.ph = PasswordHasher()


    def arg_hash(self,password):
        return self.ph.hash(password)

    def arg_ver(self,key,password):
        return self.ph.verify(key, password)
```

Figure-2 Argon2 hashing (Schlawack, n.d.)

In Stage-2 we perform AES-256 encryption on the hashed password by calculating the block size and then add padding to the password as per the block size as shown in figure-3

```
aes.py > AESCipher > __init__
import base64
import hashlib
from Crypto import Random
from Crypto.Cipher import AES

class AESCipher(object):

    def __init__(self, key):
        self.bs = AES.block_size
        self.key = key

    def encrypt(self, raw):
        raw = self._pad(raw)
        iv = Random.new().read(AES.block_size)
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        return base64.b64encode(iv + cipher.encrypt(raw.encode()))

    def decrypt(self, enc):
        enc = base64.b64decode(enc)
        iv = enc[:AES.block_size]
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        return self._unpad(cipher.decrypt(enc[AES.block_size:])).decode('utf-8')

    def _pad(self, s):
        return s + (self.bs - len(s) % self.bs) * chr(self.bs - len(s) % self.bs)

    @staticmethod
    def _unpad(s):
        return s[:-ord(s[len(s)-1:])]
```

Figure-3 Performing AES-256 Encryption (Zia, 2022)

In Stage-3 we store the encrypted hash password inside an image using LSB-image steganography which increase the security of our proposed model even further. As the data is converted into binary format before storing it inside the image in LSB steganography we add $$ sign to distinguish between the cipher text and the text of the image. So while decoding the image it becomes more easy to perform the operations. The code for the conversion and hiding of the cipher text is shown in figure-4 below

```python
def data2binary(data):
    if type(data) == str:
        p = ''.join([format(ord(i), '08b')for i in data])
    elif type(data) == bytes or type(data) == np.ndarray:
        p = [format(i, '08b')for i in data]
    return p

# hide data in given img
def hidedata(img, data):
    data += "$$"                                    #'$$'--> secrete key
    d_index = 0
    b_data = data2binary(data)
    len_data = len(b_data)

 #iterate pixels from image and update pixel values
    for value in img:
        for pix in value:
            r, g, b = data2binary(pix)
            if d_index < len_data:
                pix[0] = int(r[:-1] + b_data[d_index])
                d_index += 1
            if d_index < len_data:
                pix[1] = int(g[:-1] + b_data[d_index])
                d_index += 1
            if d_index < len_data:
                pix[2] = int(b[:-1] + b_data[d_index])
                d_index += 1
            if d_index >= len_data:
                break
    return img
```

Figure-4 Data conversion and hiding

This encrypted hash password is then store inside image using encoding techniques shown in figure-5

```python
def encode(data):
    img_name = 'bright.png'
    image = cv2.imread(img_name)
    img = Image.open(img_name, 'r')
    w, h = img.size
    if len(data) == 0:
        raise ValueError("Empty data")
    enc_img = 'bright_encry.png'
    enc_data = hidedata(image, data)
    cv2.imwrite(enc_img, enc_data)
    img1 = Image.open(enc_img, 'r')
    img1 = img1.resize((w, h),Image.ANTIALIAS)
    # optimize with 65% quality
    if w != h:
        img1.save(enc_img, optimize=True, quality=65)
    else:
        img1.save(enc_img)
```

Figure-5 Encoding of cipher text inside image

In last and final stage-4 we start the extraction of the cipher text from image by first using decoding technique of image steganography as shown in figure-6

```
# decoding

def find_data(img):
    bin_data = ""
    for value in img:
        for pix in value:
            r, g, b = data2binary(pix)
            bin_data += r[-1]
            bin_data += g[-1]
            bin_data += b[-1]

    all_bytes = [bin_data[i: i + 8] for i in range(0, len(bin_data), 8)]

    readable_data = ""
    for x in all_bytes:
        readable_data += chr(int(x, 2))
        if readable_data[-2:] == "$$":
            break
    return readable_data[:-2]

def decode():
    img_name = 'bright_encry.png'
    image = cv2.imread(img_name)
    img=Image.open(img_name,'r')
    msg = find_data(image)
    return msg
```

Figure-6 Decoding of Cipher text inside image (pranjalkalal, 2022)

Then we perform AES-256 decryption on the extracted cipher text to obtain the hash value which is compared with the original hash.

We also calculate password size, encryption time, decryption time, execution time, throughput and avalanche effect which is shown in figure-7 avalancheEffect.py > comp_count

```python
def getAEffect():
    p_t = 'karodimalpruthviraj@parakh'
    b_t=''.join(format(ord(i), '02b') for i in p_t)
    print(b_t)
    if b_t[-1] == '0':
        b = b_t[-1].replace("0","1")
    else:
        b=b_t[-1].replace("1","0")

    bin_data = b_t[0:len(b_t)-1]+b #String

    def BinaryToDecimal(binary):
        string = int(binary, 2)

        return string

    print(bin_data)

    str_data =''

    for i in range(0, len(bin_data), 7):

        temp_data = bin_data[i:i + 7]
        decimal_data = BinaryToDecimal(temp_data)
        str_data = str_data + chr(decimal_data)
    print(str_data)
    return p_t,str_data

def comp_count(p1,p2):

    print('\n\nFirst Encrypted Password:',p1)
    print('\n\nSecond Encrypted Password:',p2)
    s1=''.join(format(ord(i), '02b') for i in p1)
    s2=''.join(format(ord(i), '02b') for i in p2)
    print(len(s1))
```

Figure-7 Calculation of avalanche effect

Finally, the original password its size, encryption and decryption time, execution time, throughput is stored inside an CSV file as shown in figure-8

```python
    data['Password']= userPass[i]
    data['PassSize']=size_userPass
    data['EncryptionTime']=aes_encTotalTime
    data['DecryptionTIme']=aes_decTotalTime
    data['ExectuionTime']=executionTime
    data['Throughput']=size_userPass/aes_encTotalTime
    csvData.append(data)

from csv import DictWriter

    # list of column names
field_names = ['Password', 'PassSize', 'EncryptionTime','DecryptionTIme','ExectuionTime','Throughput']

print(csvData)
# Open CSV file in append mode
# Create a file object for this file
with open('data.csv', 'w') as csvfile:
    # creating a csv dict writer object
    writer = csv.DictWriter(csvfile, fieldnames = field_names)

    # writing headers (field names)
    writer.writeheader()

    # writing data rows
    writer.writerows(csvData)
```

Figure-8 Storing of Results in CSV file

References

pranjalkalal, 2022. Image-stegnography.
Avilable at: https://github.com/pranjalkalal/Image-stegnography
Schlawack, H., n.d. argon2-cffi: The secure Argon2 password hashing algorithm.
Avilable at: https://pypi.org/project/argon2-cffi/
Zia, M.A., 2022. Python-File-Encryptor.
Avilable at: https://github.com/the-javapocalypse/Python-File-Encryptor