National College of Ireland

Configuration Manual

MSc Research Project
Cyber Security

Virendra Yadav
Student ID: X21154384

School of Computing
National College of Ireland

Supervisor: Michael Pantridge

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Virendra Yadav
Student ID:	X21154384
Programme:	MSc in Cyber Security **Year:** 2022-2023
Module:	MSc Research Project
Lecturer:	Michael Pantridge
Submission Due Date:	15th December 2022
Project Title:	Modular approach with the blend of Argon2 Hashing and Twofish Encryption for strengthening Password Security
Word Count:	**778** **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: Virendra Yadav

Date: 15th December 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

Virendra Yadav
X21154384

1 Introduction

This manual includes all the information regarding the settings, all the prerequisites for using the proposed model, and all the applications & libraries that were extensively utilised during the model creation and execution process. The setup documentation also provides details on how to use the methods required to construct the proposed model.

2 System Configurations

2.1 Device Specification

Device:	Dell 5555 Inspiron i5 generation Laptop
Operating System:	Microsoft Windows 10 operating system
Device's RAM:	8 GB (Gigabyte)
System Configuration:	64-bit operating system (OS), x64-based Intel-processor

2.2 Software and Tools used

Operating System:	Windows 11 Home
Version of Python:	Python version 3.10.0
Code Editor used:	Visual Studio Code (VSC)

3 Installation

Installing certain Python packages is necessary in order to perform the encryption, decryption, and hashing required by the suggested models.

1. Package: Pycrypto
   Command: pip3 install pycrypto

2. Package: Pycrptodrome
   Command: pip3 install Pycrptodrome

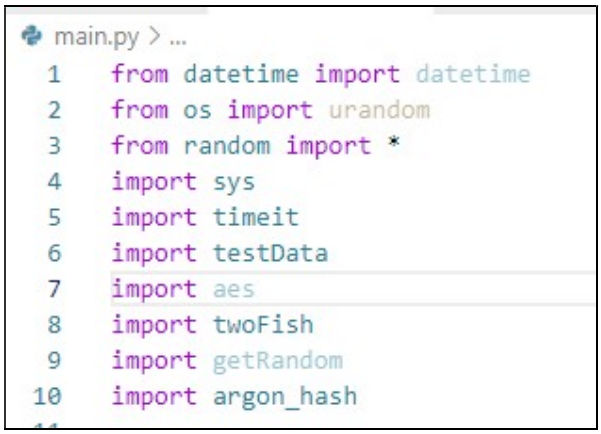
3. Package: Twofish
   Command: pip install Twofish

4 Implementation

The source code is written in Python programming language and it has been divided into 5 different code files:
These 5 files are in the same code folder, the naming of these files are as follows:

1. main.py – This file is the main file which contains the calls for all other files.
2. twoFish.py – This file contains the twofish encryption code
3. argon_hash.py – This file contains the code for Argon2 hashing
4. testData.py – This file contains all the test data coming from the main file.
5. avalance_effect.py – This file is used to calculate the avalanche effect of the model.

I. First, we have to run the main file titled as "main.py". In the main.py file the imports which were needed to be done are shown in the figure below.



II. After that we have the testData.py python file , from here we are taking the input to our the main program file (main.py)

```
testdata = [p_t,'key'
            ,str_data,'key'
            ,'aksosksjshsdkslsslg','key'
            ,'aksosksjshsdkslsslhscv','key'
            ,'aksosksjshsdkslssljsfrf','key'
            ,'aksosksjshsdkslsslksefb','key',
            'aksosksjshsdkslsslmsddfacs', 'key',
            'aksosksjshsdkslsslsdffsssdsrft','key']
```

III. Argon2 hashing is done in the next step. The password value goes through Argon2 Hashing and through Argon2 hashing we achieve the hashed value. The figure below shows the code used for the Argon2 hashing phase.

```python
from argon2 import PasswordHasher

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

    def doHashing(self,password):
        return self.ph.hash("correct horse battery staple")

    def doHashVerify(self,hash,password):
        return self.ph.verify(hash, password)
```

IV. After getting the hased value from Argon2 hashing phase, the output hashed value is feed as input to the Twofish encryption phase. (“passlib.hash.argon2 - Argon2 — Passlib v1.7.4 Documentation,” n.d.)The figure below show the code for the Twofish encryption:

```python
from twofish import Twofish

def tfencrypt(plaintext, password):

    bs = 16 #block size 16 bytes or 128 bits
    if len(plaintext)%bs: #add padding
        padded_plaintext=str(plaintext+'%'*(bs-len(plaintext)%bs)).encode('utf-8')
    else:
        padded_plaintext=plaintext.encode('utf-8')

    T = Twofish(str.encode(password))
    ciphertext=b''

    for x in range(int(len(padded_plaintext)/bs)):
        ciphertext += T.encrypt(padded_plaintext[x*bs:(x+1)*bs])
    return ciphertext

def tfdecrypt(ciphertext, password):

    bs = 16 #block size 16 bytes or 128 bits
    T = Twofish(str.encode(password))
    plaintext=b''
    for x in range(int(len(ciphertext)/bs)):
        plaintext += T.decrypt(ciphertext[x*bs:(x+1)*bs])

    text=str.encode(plaintext.decode('utf-8').strip('%'))
    return text
```

V. After encrypting the hashed value using the twofish technique the encryption time is calculated. The total execution time taken by encryption is converted into

milliseconds. (“Python Twofish Examples, twofish.Twofish Python Examples - HotExamples,” n.d.) The below figure shows the total execution time taken by the model:

```
print('Two Fish Encryption Initialisation...')
encTimeStart=timeit.default_timer()
two_fish=twoFish.tfencrypt(hashPass,key)
if j==0:
    s1=str(two_fish)[1:]

elif j==2:
    s2=str(two_fish)[1:]

print('Encrypted Password: ',two_fish)
encTimeEnd=timeit.default_timer()
encTotalTime=encTimeEnd-encTimeStart
encTotalTime*=1000
print("encTotalTime : ",encTotalTime)
print('====================================\n')
```

In order to calculate the secuirty of the model, we have written a code for avalanche calculation, which observe the number of bits flipped after password change and divide those number of bits with the total number of bits.(“Python Twofish Examples, twofish.Twofish Python Examples - HotExamples,” n.d.)

VI.

avalance_effect.py > ...

```
#p_t = '\x10\xe7\xb8\xbb\xb9j\xbd\xf1\xad\xcc\x87T\x88U\t\x06'
#p_t2 ='\xbaAkj\xb9\xdc\xf9_\x19Rk>\x81\xea|L'
def comp_count(p1,p2):

    #print(p1)
    #print(type(p1))
    s1=''.join(format(ord(i), '02b') for i in p1)
    s2=''.join(format(ord(i), '02b') for i in p2)
    #print(len(s1))
    c=0
    if len(s1)>len(s2):
        diff=len(s1)-len(s2)
        s='0'*diff
        s2=s+s2
    else:
        diff=len(s2)-len(s1)
        s='0'*diff
        s1=s+s1
    #print(s1)
    #print(s2)
    for i in range (0,len(s1)):
        if s1[i]!=s2[i]:
            c=c+1
    print(c)
    print((c/len(s1))*100)
#comp_count(p_t,p_t2)
```

VII. For the purpose of comparison we have written the code for AES, so that we can compute the values of our propsoed model (twofish encryption and arogn2 hashing) with the AES model (BoppreH, 2022).

The figure below shows the code written for AES model:

aes.py > ...

```python
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:])]
```

5 References

BoppreH, 2022. What's in the box.

passlib.hash.argon2 - Argon2 — Passlib v1.7.4 Documentation [WWW Document], n.d. URL https://passlib.readthedocs.io/en/stable/lib/passlib.hash.argon2.html (accessed 12.15.22).

Python Twofish Examples, twofish.Twofish Python Examples - HotExamples [WWW Document], n.d. URL https://python.hotexamples.com/examples/twofish/Twofish/-/python-twofish-class-examples.html (accessed 12.15.22).