Configuration Manual

MSc Research Project
Cyber Security

Somil Jain
Student ID: X21154350

School of Computing
National College of Ireland

Supervisor: Dr Rohit Verma

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Somil Jain		
Student ID:	X21154350		
Programme:	MSc in Cyber Security	**Year:**	2022-2023
Module:	MSc Research Project		
Lecturer:	Dr Rohit Verma		
Submission Due Date:	15 Dec 2022		
Project Title:	Honey2Fish - An enhanced hybrid encryption method for password and messages		
Word Count:	**724**	**Page Count: 8**	

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: Somil Jain

Date: 15/12/2022

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

Somil Jain
X21154350

1 Introduction

This manual contains details on the configuration as well as requirements for the proposed models, along with the particular libraries and needed software. Details on how to apply the techniques necessary to build the recommended model are also included in the configuration manual.

2 System Configurations

2.1 Device Specification

Processor	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
RAM	8.00 GB
System type	64-bit operating system, x64-based processor

2.2 Software and tools

OS	Windows 11 Home
Python:	Python version 3.10.9
Code Editor	Visual Studio Code

3 Installation

Some of the python packages are needed to be installed in order to do encryption and decryption.

1. Pycrypto
pip3 install pycrypto

2. Pycrptodrome
pip3 install Pycrptodrome

3. Twofish
pip install Twofish

4 Implementation

Python Code Files:
There are 5 files in the code folder.

1. main.py which is the main file containing all the calls of other files.
2. twofish.py which contains code of twofish encryption
3. honey.py which is honey encryption and decrption file.
4. testdata.py which contains the test data for the main file.
5. compare.bits.py which is used to calculate the avalanche effect.

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

```
import sys
import timeit
from datetime import datetime
import honey
import twoFish
from random import *
from pprint import pprint
import random
import testData as testData
import aes
from os import urandom
```

II. After that we from the testdata.py we have taken input in the main program file

```
testdata = ['user1',p_t,'CA','key',
            'user2',str_data,'CA','key',
            'user3','spiderman@batmad','CA','key',
            'user4','spiderman@batmal','CA','key',
            'user5','spiderman@batmae','CA', 'key',
            'user6','spiderman@batmaw','CA', 'key',
            'user7','spiderman@batmaa','CA','key']
```

III. Honey encryption is done in the step. The seed to password and the password to seed is encoded. The figure below shows the same with honeyword creation(Nguyen, 2022).

```
passwordsToSeeds[userPass] = trueSeed
seedsToMessages[trueSeed] = message
#print('sseed',trueSeed,message,passwordsToSeeds,seedsToMessages)

passwordsToSeeds[userPass + str(trueSeed - 1)] = trueSeed + 1
seedsToMessages[trueSeed + 1] = states['AL']
#print('93sseed',passwordsToSeeds,seedsToMessages)

passwordsToSeeds[userPass + str(trueSeed - 2) + "1"] = trueSeed + 2
seedsToMessages[trueSeed + 2] = states['CA']
#print('97sseed',passwordsToSeeds,seedsToMessages)

passwordsToSeeds[userPass.lower()] = trueSeed + 3
seedsToMessages[trueSeed + 3] = states['FL']

passwordsToSeeds[userPass.lower() + str(trueSeed + 1) + "3"] = trueSeed + 4
seedsToMessages[trueSeed + 4] = states['TX']
#print('104sseed',passwordsToSeeds,seedsToMessages)


passwordsToSeeds[userPass.upper()] = trueSeed + 5
seedsToMessages[trueSeed + 5] = states['TN']
```

IV. After Creating the honeyword the password is encrypted with twofish. The figure below show the same (Yin et al., 2017).

```
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])
```

V. After encrypting the password using twofish algorithm the encryption time is calculated. The time is converted into milliseconds. The figure below shows the total execution time(O, 2021).

```
print('Two Fish Encryption Initialisation...')
encTimeStart=timeit.default_timer()
two_fish=twoFish.tfencrypt(password,key)
if j==0:
    s1=str(two_fish)[1:]
    #print(s1 ,"")
elif j==4:
    s2=str(two_fish)[1:]
    #print(s2 , "")


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

VI. After encrption the password, for message retrival password is entered by the user which is shown in figure below query is hardcoded in order to reduce the user dependency.

```
# Prompt the user to crack this password
print('Decoding Password and Fetching Secret message from Honey encryption: ')
p = passwordsToSeeds.keys()
query = 'SPIDERMAN@BATMAN'
if query!=decrypt_ and query not in passwords:
    print('wrong Password...')
```

VII. For message retrival three possiblitis are shown

Correct password: True message will be print
Password matches honeyword: Fake message will print
Incoreect password: No message will print.

The figure below shows the messgae retrival code.

```
    if message in states.keys():
        if m != trueSeed:                     # Honey checker
            print("Intruder! Sound alarm!")
            print('Fetching Secret Message from Honey Encryption: ',seedsToMessages[keySeed])

    # Check seeds
        else:
            #print('Password Decrypted: ',query)
            print('Fetching Secret Message from Honey Encryption: ',states[seedsToMessages[m]])
            print('=====================================\n')
    else:
        if query == password:
            #print('Password Decrypted: ',query)
            state=random.choice(list(states.keys()))
            print('Fetching Secret Message from Honey Encryption: ',message)
            print('=====================================\n')
        else:
            print("Intruder! Sound alarm!")
            print('Password Decrypted: ',query)
            state=random.choice(list(states.keys()))
            print('Fetching Secret Message from Honey Encryption: ',states[state])
except KeyError:
```

VIII. For evaluation purpose Avalanche Effect is calculated. The file compare_bits.py has the code for Avalanche Effect. The figure shows the same.

```
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)
```

AES is also implmented for the evalution and comparsion of AES and twofish. The code of the same is shown below (Campos, 2021).

```
from os import urandom
from Crypto.Cipher import AES


def Enc_AES (password , key , iv):
    # For Generating cipher text
    obj = AES.new(key, AES.MODE_CBC, iv)
    # Encrypt the message
    #print('Original message is: ', password)
    encrypted_text = obj.encrypt(password.encode('utf8'))
    #print('The encrypted password', encrypted_text)
    return encrypted_text

def Dec_AES (enc_password ,key, iv):
    # Decrypt the message
    rev_obj = AES.new(key, AES.MODE_CBC, iv)
    decrypted_text = rev_obj.decrypt(enc_password)
    print('The decrypted text', decrypted_text.decode('utf-8'))
    return  decrypted_text
```

5 References

Campos, P.T., 2021. AES Implementation in Python. Quick Code. URL https://medium.com/quick-code/aes-implementation-in-python-a82f582f51c2 (accessed 12.15.22).

Nguyen, V., 2022. Honey Encryption.

O, K., 2021. Python TwoFish Encryption. DevRescue. URL https://devrescue.com/python-twofish-encryption/ (accessed 12.15.22).

Yin, W., Indulska, J., Zhou, H., 2017. Protecting Private Data by Honey Encryption. Security and Communication Networks 2017, e6760532. https://doi.org/10.1155/2017/6760532