

Configuration Manual

MSc Research Project Cyber Security

Taraka Raghavendra Sai Panchakarla Student ID: X22150951

School of Computing National College of Ireland

Supervisor: Dr Rohit Varma

National College of Ireland



MSc Project Submission Sheet

School of Computing

Student Name: Taraka Raghavendra Sai Panchakarla

Student ID: x22150951

Programme: MSc in Cyber Security

Module: MSc Research Project

Lecturer: Dr Rohit Varma Submission Due

Date: 25 April 2024

Project Title: Secure Communication Using Quantum Key Distribution and Honey Encryption

Page Count: 9

Year: 2023-2024

Word Count: 718 excluding references and appendices

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.

<u>ALL</u> 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: Taraka Raghavendra Sai Panchakarla

Date: 25 April 2024

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

Taraka Raghavendra Sai Panchakarla Student ID: x22150951

1. Introduction

This handbook includes specifics about the setup and prerequisites for the suggested models, in addition to the required software and libraries. The configuration handbook also contains instructions on how to use the methods required to construct the suggested model.

2. System Configurations

Research Design and Approach

Processor12th Gen Intel(R) Core (TM) i7-12700H2.30 GHzRAM16.0 GB (15.7 GB usable)System type64-bit operating system, x64-based processor

Software and tools

OS Windows 11 Home Single Language Python Python 3.12.1 Code Editor Jupyter Notebook

3. Installation

Some of the python packages need to be installed like pip install python and pip install jupyter and next install pip jupyter notebook.

Microsoft Mindows (Nersion 18.8.2263).3447
(c) Microsoft Carporation. All rights reserved.
C:\Users\Tarak\OneDrive\Desktsp\Research>jupyter notebook
[1 2024-04-10 10:57:20.199 Serverkop] Extension package jusyster.lsp took 0.1420s to import
1 2024-04-10 19:57:20.387 ServerApp Extension package jupyter_server_terminals took 0.1937s to import
1/2 2024-06-10 19-57:20.666 Serverke0 A 'juprter server extension points' function mas not found in notebook shim. Instead, a ' juprter server extension paths' function mas found and will be used for nom. This function name will be degregated in future releases of Ju
prist Server.
[2024-04-10 19:57:20.663 ServerApp] jupyter_lsp extension was successfully linked.
1 2029-0 10 55720.605 prevedpoj preve stor i eternisto mas processivally linked.
1 202+0-10 15:77:20:00 StreveRpG [uptrick] attributes and streveRpd [uptrick] attributes attributes [united to attributes attributes]
3 202-00-10 35:57:30.070 ServerApp Intelson is successfully linked.
1 2024-06-10 19:57:21.830 Serverkagi notebook.shin extension mas successfully linked.
1 2024-04-10 19:57:21.013 Serverbag notebook.shin extension mas successfully loaded.
[2 202+0* 19 57:2:01 Streven(b)] united to a successfully loaded.
1 202+0-0-10 1/3722.103 Serverkoj jugita. Log i extension mas successfully loaded. 1 202-0-0-10 1/3722.103 Serverkoj internetinals i entension mas successfully loaded.
1 202-0-10 15:57:21.390 Lisbop) Juppite artension loaded from (:\User\Law\Desitop) Rescarch\Lib\site-package\jupyterlab
2 202-0-0-10 19:57:22: 200 Labop) apprentiate on directory is C. Users Varak Monte Public Control Cont
a data wa wa 1975 22,298 wanta wa kata wa kata wa kata kata kata kat
1 2020 00 19 5977 2020 Exercised Example Extension Example 2 Second State 1 2020 00 10 19 59772 2020 Exercised a second State 1 2020 00 10 19 59772 2020 Exercised a second State 1 2020 00 10 10 10 10 10 10 10 10 10 10 10 10
[2 202+49-10 15/5722.209 ServerApp] Jupper and [vertexistor max successfully] London. [2 202+49-10 15/5722.209 ServerApp] Jupper and [vertexistor max successfully] London.
1: Jogeneral 19:57:21:00 Serverbag, Intension mas successfully Loases. 1: 2020-0-19:15:77:21:08 Serverbag, Intension France, Classes/Landon/Prise/Destato/Research
Is deed to by provide settering increases rate of the other setter
1: 000000 12:07:22.070 380000000 000000 381000000000000000000
1 282-49-10 15:57:21. 920 ServerApp http://127.8.8.1.8888/treaf/dom/75665336/02/26465392446018e=1dafec/dd592446
1: 2020-0-10 15/5727.20 Serverkpg) (a CU)/14/70.1.0000/1000/10000-200004070/0005/00010470(005/0000)
La defeneral association prevented in 15/22.2097 Serverbal
[L 2029 99 30 12 17:07:02:77) 38290(0)[0]
To access the server, open this file in a browner;
(i) inclusion of the set of th
Table 17 Sec 1995 21 Hadden dynamia ag ywy cert a minawy ywy cert a minawy ywy cert a minawy dynamia a minawy ywy cert a minawy ywy cert a minawy dynamia a minawy
0 C kgy win parket with 0 C kiele with a http://b.ck.hnstlabs/reservice/win=756663a/a/a a 32c52a6#3342a6#31944idafec76d1920#fd
ns.q.//uckations.com/ctavit.com/=commons/actualized/actua
mup://art.org.ac.mom/uwer.dom-noncasterezizaanesizazaane
is accessed in a constrained and the second and the second and the second accession access
resters, sit tangang server, torkan, typestilp tangang server, miler tangang server, solve tangang server tan, victor functiongang server tan, victor tangang server tan, setter 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
0-ras ⊃ bedoget entrange. At setem tend tracen monocks and being mode, mixth may 0.68 = make this debugger miss breakpoints. Please pass * thread mode, mixth may
0.886 - to produce the second se
9.000 - to pytom to calante travem montes. 0.000 - Note: Debugging will proceed. Set PMOIND.DISABLE_FILE_VALDATION=1 to disable this willdation.
View - write lowdyng aist proteins at rotung lands. Julg william the land that it is a standard this valuation. [1 2024-0-10 15:502.104] Struttering Wennig Lattici 120446-013-900-0145-01400
12 days of up 13 double-show have very and a second very mark water second very and second very se
0.005 - usedages matiming, it seems that revent motives are being tools, mich may 0.005 - make the debugger matismic streakpoints. Please pass - Stream, motive and f
0.005 - make the decogner mais transports, reasks pairs "Artern_Boolutes-off"
0.009 - 10 pytom to Guadate Tracem Boales. 0.009 - Nets Debugging will proceed. Set PMORED.ISABLE_FILE_VALIDATION=1 to disable this validation.
0.emp - matter toetuuggung mitt proceedes - ster voorschaftende, mitteen toetung -
12 July 4 w 10 21 July 22 July
12 - 2024 The Tay Tay South And Tay Ten Market State
0.ems - Jennagyer matrixing: it seems tank trizen moutes att being unon, mitch may 0.089 - make the debugger mits preaksparts. Trizen "molte-may off
0.000 - many two developer has received as Articen_modules-orr
0.005 - to Synche to mission Proceed. Set PUDNO.DISABLE_FILE_VALDENTON: to disable this validation.
n ens - mate: useugging mil proteens - set vulux_linamil_rill_null_null_milient thi valuation. [1] 2024-0-10 15:05:55.35 as yearchol Connecting to kernel "Mission-Dirac Default-default-default-default-defau
1: 2024-91 19:59:51.33 Serverxeg: Connecting (on network: 1900HU): -000FWU: -000FW
1: 2024/#1.2021/2012/2012/2012/2012/2012/2012/201
1: adapted to doctationed serverings) saving title at / (pow law minety) introduction .jpph [1: adapted to 28:09:37:09 Servering) Saving of Ea at / (DK and Honey Encryption .jpph)
1: adv=== 10 adv
La deverse la varia ve las serventos sur la varia de la varia mane un concession de la varia
1: azerweria azerwering) asarda rite az vigik ade mene sinceptian izgen [1: azerweria zalokini, zaj szenken] asarda rite intergetan izgen a
1: adarwer ta arten wurdt Sarverken) Saving Tile at 700 kan ennen stortyminen Japan 1: adarwer ta 2016012: add Saverken Saving Saving Tile at 700 kan ennen stortyminen Japan
1 adverver ta an of 1.00 security barray rates at values and the security time types of the security table at
1: addrefer 10 addocts.com bereverkog Saving File at / QWA and Inney Encryption Japon
It says as to serve be served by the art show one wind the best between the

Figure 1: Opening Jupyter Notebook

Download Microsoft	ft Teams Desktop 🖸 Home × 🧧 QDK and Honey Encryption 📿 Home +					- 0	
> e	Iocalhost:8888/tree	L.	V 🖉	c.	L 🗆 (4
	💭 Jupyter						
	File View Settings Help						
	Files O Running						
	Open Download Rename Duplicate Delete		∗ New 🛓	Upload C			
	🗉 Name	_ L	ast Modified	File Size			
	🗆 🖿 DLLs	2	months ago				
	🗆 🖿 Doc	2	months ago				
	🗆 🖿 etc	2	months ago				
	🗆 🖿 include	2	months ago				
	🗆 🖿 Lib	2	months ago				
	🗆 🖿 libs	2	months ago				
	🗆 🖿 New folder		2 days ago				
	🗆 🖿 pro jup	2	months ago				
	🗆 🖿 Scripts		8 days ago				
	🗆 🖿 share	2	months ago				
	🗆 🖿 tcl	2	months ago				
	🗆 🖿 Week 1		last month				
	🗆 🖿 Week 2		last month				
	🗆 🖿 Week 3		9 days ago				
	🗆 🖿 Week 4		last month				
	🗆 🖿 Week 8	58 r	minutes ago				
	🗆 * 📕 QDK and Honey Encryption.ipynb	58 r	minutes ago	254.9 KB			
	🖌 • 🖪 xipynb		months ago	11.2 KB			
		4	months ago	36 KB .	-		

4. Implementation

Figure 2: Jupyter Notebook

Python Code: There are seven major steps in the Python code.

- 1) The performing of Quantum Key Distribution (QKD). This has three more steps as follows:
 - First step contains code used for generating Qubits.
 - Next step will be consisting of the program for comparing of the qubits generated.
 - The third step will be of generating of key which is used to encryption of the secret key.
- 2) The second phase will be execution of Honey Encryption (HE) and Decryption. This will be having two steps as follows:
 - Once the secret key is generated that needs to be encrypted and protecting it with a password. This encryption is the fourth step.
 - The next step will be to decrypt the encrypted secret key to the legitimate source user.
- 3) The sixth step will running the main function program contains the commands and displaying all the results as per the program requirements.
- 4) The last step will be showing the performance and Evaluation reports such as graphs and other things.

1.

As we mentioned above the first step for generating the qubits is to run the below code shown in Figure 3. As per BB84 protocol Qubits are generated with bits and bases as per Table 1. if the rectilinear basis (y) compared with bit value 0 it is called 0_p. else if rectilinear basis (y) compared with bit value 1 it is called 45_p. else if diagonal basis (x) compared with bit value 0 it is called 90_p. else it is called as 135_p respectively as per the input values (Anusuya Devi V, 2021).

BB84 Protocol Polarization Scheme					
Rectilinear Polarization Basis	0_p	45_p			
(Y)	_				
Diagonal Polarization Basis	90_p	135_p			
(X)		_			
Bit Value	0	1			

Table 1: BB84 polarization scheme

```
def generate_qubits(bits, basis):
    qubits = []
    for bit, base in zip(bits, basis):
        if base == 'y':
            if bit == '0':
                qubits.append('0_p')
        else:
                qubits.append('90_p')
        elif base == 'x':
            if bit == '1':
                qubits.append('135_p')
        else:
                qubits.append('45_p')
        return qubits
```

Figure 3 Code for generating Qubits

2.

Once the Qubits of A and B are formed, they much be compared if all qubits of A and B are the same print as open the message. else if no qubits are the same print do the generation of qubits again. else if some quits are same and some or not same then perform an XOR operation between matched bits value and A remaining not matched bits. The code for this will be as shown in Figure 4 (Agarwal, 2023).

```
def compare_qubits(qubits_a, qubits_b):
    if len(qubits_a) != len(qubits_b):
        return "Qubit lists have different lengths."
    matching_count = 0
    secret_key = ''
    for a, b in zip(qubits_a, qubits_b):
        if a == b:
            matching_count += 1
        else:
            secret_key += str(int(a.split('_')[0]) ^ int(b.split('_')[0]))
    if matching_count == len(qubits_a):
        return "Message for B: All qubits matched."
    elif matching_count > 0:
        return "Secret Key: " + secret_key
    else:
        return "Intruder Alert."
```

Figure 4 Code for comparing the Qubits

3.

If some quits are same and some or not same, then perform an XOR operation between matched bits value and A remaining not matched bits. A key need to be used for hashing and password so that for decryption. The code for creating the key is shown in Figure 5.

```
def generate_key(password, salt):
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        iterations=100000,
        salt=salt,
        length=32,
        backend=default_backend()
    )
    Figure 5 Congreting of key
```

Figure 5 Generating of key

4.

Once all the things such as bits, bases, qubits, secret key, key, and password provided by using these the secret key need to be encrypted. That encryption will be done by using code shown below Figure 6 (Nahri Syeda Noorunnisa, 2016).

```
def encrypt_message(message, key):
    cipher = Fernet(key)
    start_time = time.time()
    encrypted_message = cipher.encrypt(message.encode())
    end_time = time.time()
    encryption_time = end_time - start_time
    return encrypted_message, encryption_time
```

Figure 6 Code for encryption of secret key

5.

The encrypted message needs to be decrypted only for the receiver. Do so the receiver need to give the right password kept by the sender to decrypt and know the secret key. This type of code shown in Figure 7 (Campos, 2021).

```
def decrypt_message(encrypted_message, key):
    cipher = Fernet(key)
    start_time = time.time()
    decrypted_message = cipher.decrypt(encrypted_message).decode()
    end_time = time.time()
    decryption_time = end_time - start_time
    return decrypted_message, decryption_time
```

Figure 7 Code for decrypting

6.

The six step we have the main function where it contains the commands and displaying all the results as per the program requirements. The code is shown in Figure 8.

```
def main():
   # Eavesdropping attack detection
   eavesdropping_detected = False
   if input("Is there an eavesdropping attack? (True/False): ").lower() == 'true':
       print("Eavesdropping attack confirmed. Program execution stopped.")
       return
   # Secret key generation
   user_password = input("Enter the password to secure the secret key: ")
   salt = b'\x9e\xe1\xea\xb9\x05~\x8b\x06\x97\x8fI!\x89r\xb9\x12'
   user_length = int(input("Enter the length of the bits: "))
   basis_a = ''.join(random.choice(['x', 'y']) for _ in range(user_length))
   bits_a = ''.join(str(random.randint(0, 1)) for _ in range(user_length))
   basis_b = ''.join(random.choice(['x', 'y']) for _ in range(user_length))
   bits_b = ''.join(str(random.randint(0, 1)) for _ in range(user_length))
   print("User A Basis:", basis_a)
   print("User A Bits:", bits_a)
   print("User B Basis:", basis_b)
   print("User B Bits:", bits_b)
   qubits_a = generate_qubits(bits_a, basis_a)
   qubits_b = generate_qubits(bits_b, basis_b)
   print("User A Qubits:", qubits_a)
   print("User B Qubits:", qubits_b)
   comparison_result = compare_qubits(qubits_a, qubits_b)
   print(comparison_result)
   # Encryption and decryption of secret key
   user_message = input("Enter the message to be encrypted with the secret key: ")
   key, key_generation_time = generate_key(user_password, salt)
   print("Secret Key:", key)
   print("Time taken to generate key:", key_generation_time, "seconds")
   encrypted_message, encryption_time = encrypt_message(user_message, key)
   print("\nEncrypted Message:", encrypted_message)
   print("Time taken for encryption:", encryption_time, "seconds")
   entered_password = input("Enter the password to decrypt the message: ")
   entered_key = generate_key(entered_password, salt)[0]
   if entered key == key:
       decrypted_message, decryption_time = decrypt_message(encrypted_message, key)
       print("Decrypted Message:", decrypted_message)
       print("Time taken for decryption:", decryption_time, "seconds")
    else:
       print("Intruder alert! Incorrect password.")
       # Provide a random key to A user
       random_key = generate_key(str(random.randint(0, 999999)), salt)[0]
       print("Random Key provided to A user:", random_key)
```

Figure 8 Main functions

7. last step will be showing the performance and Evaluation reports such as graphs

```
# Generate graphical representation of total execution according to bytes vs time
X = []
y = []
start_time = time.time()
total_bytes = 0
for _ in range(1000):
    key, key_generation_time = generate_key('password', salt)
    encrypted_message, encryption_time = encrypt_message('message', key)
    decrypted_message, decryption_time = decrypt_message(encrypted_message, key)
    total_bytes += len(key) + len(encrypted_message) + len(decrypted_message)
    elapsed_time = (time.time() - start_time) * 1000
    x.append(elapsed time)
   y.append(total_bytes)
plt.plot(x, y)
plt.xlabel('Time (milliseconds)')
plt.ylabel('Total Bytes Exchanged')
plt.title('Total Bytes vs Time')
plt.show()
# Generate graph for encryption time
x_encryption = []
y_encryption = []
for _ in range(1000):
    _, encryption_time = encrypt_message('message', key)
    x_encryption.append(encryption_time * 1000) # Convert to milliseconds
    y_encryption.append(len(encrypted_message)) # Length of encrypted message
plt.plot(x_encryption, y_encryption)
plt.xlabel('Time (milliseconds)')
plt.ylabel('Encrypted Message in MB')
plt.title('Encryption Time vs Encrypted Message')
plt.show()
# Generate graph for decryption time
x_decryption = []
y_decryption = []
for _ in range(1000):
    _, decryption_time = decrypt_message(encrypted_message, key)
    x_decryption.append(decryption_time * 1000) # Convert to milliseconds
    y_decryption.append(len(decrypted_message)) # Length of decrypted message
plt.plot(x decryption, y decryption)
plt.xlabel('Time (milliseconds)')
plt.ylabel('Decrypted Message in MB')
plt.title('Decryption Time vs Decrypted Message')
plt.show()
```

Figure 9 Performance and Evaluation Graphs

5. References

V, A.D. and V, K. (2021) Enhanced BB84 Quantum Cryptography Protocol for secure communication in Wireless Body Sensor Networks for medical applications, Personal and ubiquitous computing. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7971400/ (Accessed: 20 April 2024).

Agarwal, Neha & Verma, Vikas. (2023). Comparative Analysis of Quantum Key Distribution Protocols: Security, Efficiency, and Practicality. 10.1007/978-3-031-48774-3_10.

Noorunnisa, N.S. and Siddiqui, R.A. (2016) review on Honey Encryption Technique, International Journal of Science and Research. Available at: https://www.researchgate.net/publication/296806881_Review_on_Honey_Encryption_Techn ique (Accessed: 20 April 2024).

Campos, P.T. (2021) AES implementation in Python, Medium. Available at: https://medium.com/quick-code/aes-implementation-in-python-a82f582f51c2 (Accessed: 20 April 2024).