Enhancing the Security of Public Key Cryptography in Addressing Quantum Computing Threats

1st Oluwole Oguntoyinbo
School of Computing and Eng
University of West London
London, United Kingdom
21516296@student.uwl.ac.uk

1st Abel Ofori-Yeboah
School of Computing and Eng
University of West London
London, United Kingdom
Abel.yeboah-ofori@uwl.ac.uk

2nd Chukuemeka Ikenga-Metuh
School of Computing and Eng.
University of West London
London, United Kingdom
32110022@student.uwl.ac.uk

3rd Yogesh Ghimire
School of Computing and Eng
University of West London
London, United Kingdom
21560901@student.uwl.ac.uk

4th Efua Obode
School of Computing and Eng
University of West London
London, United Kingdom
21582642@student.uwl.ac.uk

5th Adrian Brissett
School of Computing and Eng
University of West London
London, United Kingdom
32138694@student.uwl.ac.uk

Abstract—**The potential of quantum computing to compromise traditional public-key cryptography (PKC) has led to a pressing need for more efficient and resilient cryptographic methods. Recent advancements, such as the 800x improvement in the logical error rate by Microsoft and QUANTINUUM, underscore the urgency of this issue. Post-quantum cryptography (PQC) offers a promising solution designed to withstand quantum attacks. This paper examines how PQC and other advanced cryptographic techniques can enhance the security and efficiency of PKC in the quantum era. Current PKC methods, despite revolutionizing secure online communication, are vulnerable to flaws in their underlying algorithms. The proposed hybrid model integrates PQC with traditional PKC, leveraging both technologies' strengths to mitigate quantum threats while preserving established security benefits. The investigation addresses critical research questions: how to enhance PKC key management for better efficiency and security, and which PQC techniques can protect PKC from quantum attacks. This paper's contribution includes identifying vulnerabilities in existing PKC, comparing RSA and post-quantum algorithms, and implementing a PQC algorithm between a client and server to analyze integration with existing protocols. Finally, the paper recommends security mechanisms to improve PQC protocols, ensuring robust protection against emerging quantum computing capabilities.**

Keywords—Quantum Computing, Public Key Cryptography, Post Quantum Cryptography, Cyber Security

I. Introduction

The evolution of public-key cryptography has significantly transformed modern communication and ensured the security of online transactions. Despite its profound impact, public-key cryptography encounters various challenges that may hinder its effectiveness and security. These limitations, as outlined by Xie et al. [1], include the complexities associated with key management in securely creating, distributing, storing, and revoking keys essential for utilizing public-key cryptography. The integrity and authenticity of keys are guaranteed through effective key management, a critical component of public-key cryptography. However, managing numerous keys within a large system can prove arduous and inefficient, amplifying concerns regarding key availability and confidentiality, as noted by Juniawan [2]. The security of public-key cryptography faces a significant threat from quantum computing advancements, which may accelerate the resolution of mathematical challenges such as computing discrete logarithms and factoring large numbers crucial to many public-key cryptosystems. This development enables quantum attacks to potentially compromise classic public-key cryptosystems such as RSA and elliptic curve encryption (ECC) as stated by Wang et al [3].

Achieving quantum cryptographic security using mathematical problem-solving is considered challenging, even for quantum computers. Transitioning to post-quantum cryptography is a complex process that requires careful planning and coordination. Post-quantum cryptography will also become the standard for cryptographic protocols in the future, such as:

- Lattice-based Cryptography
- Code-based Cryptography
- Hash-based Cryptography
- Multivariate Cryptography

Additionally, the computational complexity embedded in public-key cryptography, depending on complex mathematical dilemmas such as computing discrete logarithms and factoring extensive numbers, is progressively open to security breaches as computational capabilities rise, decreasing the overall security of public-key cryptosystems. The high computational cost associated with public-key cryptography techniques may render them ineffective and sluggish, particularly in resource-constrained environments[4]. Side-channel attacks manipulate the physical attributes of cryptographic systems, encompassing electromagnetic radiation, power consumption, and timing, to covertly obtain sensitive information, leading to a security risk for public-key cryptography. Safeguards are imperative to thwart such attacks and uphold the integrity of public-key cryptosystems. Homomorphic encryption presents a promising approach, allowing computations on ciphertexts without prior decryption. Nonetheless, its computational demands and costliness hinder its practicality and efficacy [5]. RSA methods are vulnerable to attacks such as interception, interruption, modification, and fabrication, highlighting the importance of secure implementations [6]

Researchers advocate for various strategies like key encapsulation mechanism, quantum key distribution, post-quantum cryptography, and homomorphic encryption to bolster the effectiveness and security of public-key cryptography, ensuring its continued relevance in contemporary communication and secure online transactions. This necessitates further research and experimentation to evaluate and compare these proposed methods with existing practices, aiming to optimize public-key cryptography for broader applications in e-commerce, cloud computing, and secure communication. The contribution of this paper is threefold. The first objective is to explore existing literature and current advancements in the field of cryptography to identify and analyze the weaknesses and vulnerabilities of prevalent public key cryptography. Further, we compare encryption algorithms such as RSA to determine secure features and relevance compared to post-quantum algorithms. Furthermore, we implement a post-quantum cryptography (PQC) algorithm between a client and server and analyze the differences with PKC and how PQC integrates with the existing protocols. Finally, we recommend security mechanisms to improve PQC protocols.

II. Related works

A. Overview

This section discusses the state-of-the-art post quantum cryptography, related works, and the existing cryptography approaches. Encryption techniques like RSA and AES are known as classical computing and can be used to secure cloud computing and IoT for big data processing [7]. Further, RSA encryption in VPN environments enhances data security by establishing secure tunneling [8]. Quantum-Resistant Cryptography - Recognizing the threat of quantum computers to existing public-key algorithms, recent works by Borges et al [9] delve into quantum-resistant cryptography, exploring new cryptographic primitives that remain secure even in the era of quantum computing. Post Quantum Cryptography typically considers public-key cryptography methods designed to resist quantum computer-based attacks.

B. Homomorphic encryption.

Homomorphic encryption enables calculations to be made on ciphertexts without first having to decrypt them. However, homomorphic encryption is computationally resource-intensive and expensive, making it ineffective and difficult to use in actual practice [10]. In abstract terms, the homomorphic encryption (HE) method facilitates arithmetic operations on encrypted data. Nonetheless, the method's effectiveness is constrained due to its inefficiency in executing specific functions, particularly those involving multiple multiplications. To address this issue, [11] introduced a novel HE-based secure computation approach known as Homomorphic Encryption for Stochastic Computing (HESC); this innovative scheme enables homomorphic evaluation of both stochastic addition and multiplication operations, eliminating the need for bootstrapping. The HESC framework is established on additive/multiplicative HE, which exclusively supports homomorphic addition/multiplication, thereby achieving homomorphic evaluation of stochastic multiplication. Leveraging stochastic computing (SC) characteristics, the HESC incorporates stochastic operations where random multiplexing and bit-parallel logic operations are utilized for stochastic additions and multiplications. Initially, [11] outlines a fundamental HESC scheme grounded in additive/multiplicative HE, followed by introducing an efficient HESC scheme capitalizing on lattice-based cryptography's parallelism (specifically, plaintext packing and vectorized homomorphic evaluation). A novel stochastic addition operation is also presented in this research, enhancing the accuracy of the HESC but leading to larger ciphertext sizes. Consequently, [11] proposes a method to reduce ciphertext sizes while preserving the scheme's accuracy. The initial performance evaluation of the HESC utilizing various HEs is showcased, illustrating its utility in polynomial functions and oblivious inference within a neural network. Ultimately, the results highlight the advantages of the proposed scheme over conventional approaches, prompting [11] to discuss potential implications and research avenues for HESC within the realms of cryptography and HE implementations. Efficiency Improvements [12] explored techniques to enhance the computational efficiency of public-key cryptography. This includes optimized algorithms, parallelization, and hardware acceleration.

C. Quantum Key Distribution.

A proposed advanced architecture was introduced to address the challenges of public-key cryptography by utilizing SimuloQron, a quantum network simulator enabling the development and testing of quantum communication protocols in a simulated setting. This architecture implements the BB84 protocol for quantum key distribution (QKD) and one-time pad (OTP), effectively thwarting potential eavesdroppers from engaging in malicious activities within the communication channel and cyber domain. The security measures are upheld by safeguarding the public key and maintaining its state through applying quantum cryptography and the BB84 protocol, which involves the utilization of polarized photons and randomly selected bases by the communicating parties, Alice and Bob, to establish a cryptographic key securely. The security of this protocol relies on the tenets of quantum mechanics, where any interception attempts by an eavesdropper named Eve would introduce detectable errors. The study was conducted by [13]. [14] highlight a system that utilizes post-quantum cryptography (PQC) and public key infrastructure (PKI) to secure productive and secure confirmation within Quantum Key Distribution (QKD). This strategy aims to decrease the requirement for symmetric keys in authenticating a QKD network, thereby streamlining the process. Experimental Verification: The study validates experimentally the feasibility, effectiveness, and consistency of the PQC algorithm in authenticating QKD. The article illustrates the practical implementation of this authentication approach by illustrating the benefits of PQC during the onboarding of new users to the QKD network. Enhanced Security: Through integrating QKD with PQC authentication, the article seeks to boost the security of quantum-safe communication. It underscores the significance of short-term security assumptions about the PQC algorithm in ensuring the long-term security of distributed keys, providing a promising strategy for secure key exchange in the age of quantum computers.

D. Extrapolated Dihedral Coset Problem(EDCP)

[15] introduced a quantum public-key encryption system founded on the Extrapolated Dihedral Coset problem (EDCP), which is equivalent, through quantum polynomial-time reductions, to the Learning with Errors (LWE) issue. The proposed system guarantees information-theoretical security for a restricted number of public keys (approximately linear concerning the security parameter). If the number of public keys is polynomial, the complexity of breaking the system becomes as challenging as solving the LWE problem. In design, the public keys are quantum states comprising O~(n) qubits in size. The key generation and decryption processes necessitate O~(n) qubit operations, while the encryption process involves O (1) qubit operations, making the process efficient. Quantum key distribution protocols are susceptible to side-channel attacks that exploit the temporal difference in detector responses utilized for acquiring key bits. The conventional recommendation to counter this timing side-channel attack involves utilizing a wider time bin width rather than high-resolution timing data. It is commonly believed that employing a larger bin width diminishes the resolution of

detector responses, thereby purportedly reducing information leakage to potential eavesdroppers. However, [16] challenges this traditional belief by illustrating that increasing the bin width does not consistently decrease the mutual information between key bits and an eavesdropper's observation of detector responses. Instead of arbitrarily widening the bin, careful selection is advocated due to the fluctuation of mutual information about the bin width. Furthermore, an analysis of the impact of full-width half maximums (FWHMs) of detector responses on mutual information reveals that decreasing the FWHM leads to increased mutual information. Lastly, the initiation time of binning is emphasized as a crucial factor in the binning process, with mutual information exhibiting periodic fluctuations relative to it. [17] highlighted the importance of addressing challenges such as the development of resilient Quantum Key Distribution (QKD) protocols and secure post-quantum cryptographic algorithms, which are crucial for fully leveraging the benefits of quantum cryptography and the threats that come along with it. Efforts in research were focused on seeking efficient solutions tailored for Internet of Things (IoT) devices, a necessary element for the practical implementation of quantum cryptography applications, including quantum cryptocurrency.

E. *Multivariate Polynomial Public Key Digital Signature.*

The research by [18] presents the MPPK/DS Algorithm, which introduces the Multivariate Polynomial Public Key Digital Signature (MPPK/DS) algorithm, developed with the aim of being quantum-resistant and capable of withstanding diverse attacks. The Enhanced Security Feature of MPPK/DS manifests in its ability to ensure security by complicating the process of deriving relationships between the private key components, thereby bolstering the overall security posture. [19] make a significant contribution through an in-depth analysis of the effects of quantum computing on public key infrastructures (PKIs) utilized in operational settings, underscoring the pressing necessity for a prompt shift towards post-quantum remedies to counteract quantum-related threats.

F. *Public Key Infrastructure.*

The scholarly work by [19] expands the existing literature by scrutinizing the security concerns associated with a particular PKI framework, such as infiltrating trusted certificates and furnishing pragmatic security recommendations to fortify defenses against quantum assaults. The research's emphasis on the susceptibilities of public-key cryptographic techniques to Shor's algorithm and the imperative nature of substituting current cryptographic primitives supplements the current body of knowledge on the critical nature of modernizing cryptographic algorithms for quantum resilience. By exploring the viability of post-quantum algorithms in security infrastructures for operational contexts and tackling obstacles in the implementation of quantum key distribution technology, the manuscript provides valuable perspectives to the literature on pragmatic considerations for deploying quantum-secure solutions in genuine scenarios.

G. *Standardization.*

Both quantum and post-quantum cryptography undergo active standardization processes. In particular, standardization of the QKD technology is considered by several agencies, such as ETSI and ISO. Post-quantum cryptography standardization is centered around NIST (National Institute of Standards and Technology). NIST has realized the need to deploy algorithms to counter the possible risks presented by quantum computers, particularly in the domain of public key cryptography. NIST has identified a variety of algorithms to be included in their post-quantum cryptography initiative. Four algorithms have been selected. CRYSTALS-Kyber has been chosen for key encapsulation, whereas CRYSTALS-Dilithium, FALCON, and SPHINCS+ have been designated for digital signatures. Although algorithms such as BIKE, Classic McEliece, HQC, and SIKE were evaluated, NIST has not met the standardization criteria [20].

III. Methodology

This section presents an overview of the approach used for the paper. We used a qualitative approach to evaluate the security and efficiency of public key cryptography algorithms in the context of quantum threats. The approach was divided into four main phases: modelling, implementation, benchmarking, and security analysis. The method was chosen for its suitability in exploring complex phenomena and gaining a deep understanding of PKC/PQC technologies. All implementations and simulations were conducted in a controlled environment, ensuring no real user data was used or compromised.

A. *Data Collection Methods*

Document Analysis: Review academic literature, industry reports, and technical documentation related to PKC/PQC algorithms, protocols, implementations, and security standards. This analysis aims to understand existing practices and identify areas for improvement comprehensively. This analysis was done using the qualitative method. Cryptanalysis: Employing cryptanalysis techniques to assess the security of PKC/PQC algorithms and protocols. This involves systematically analyzing cryptographic vulnerabilities, potential weaknesses, and attack vectors to inform recommendations for enhancing PKC security. This analysis was done using the quantitative method.

B. *Data Sources*

Leveraging academic resources, including books, journals, and scholarly papers, accessible via the UEL library portal. Utilizing sources such as IEEE Xplore, ACM Digital Library, and Google Scholar to ensure a diverse and scholarly foundation for analysis.

C. *Rationale for Approach.*

The chosen methodology follows a structured and systematic approach to address the aims and objectives of the research. By starting with a narrative systematic literature review, we establish a solid foundation of existing knowledge and identify gaps in current research. The vulnerability assessment using Python cracking tools allows us to understand the vulnerabilities of traditional PKC algorithms in relation to quantum attacks. Implementing post-quantum cryptographic algorithms using PQ Clean Python libraries enables us to explore practical solutions for resisting quantum threats. Key encapsulation mechanisms are essential for ensuring the long-term security of cryptographic systems, especially in the context of quantum-safe solutions. The experimental evaluation provides empirical evidence to validate the effectiveness and feasibility of the proposed cryptographic solutions in real-world scenarios. Overall, this methodology enables a comprehensive analysis of security enhancement techniques in the face of emerging quantum technologies.

IV. Implementation

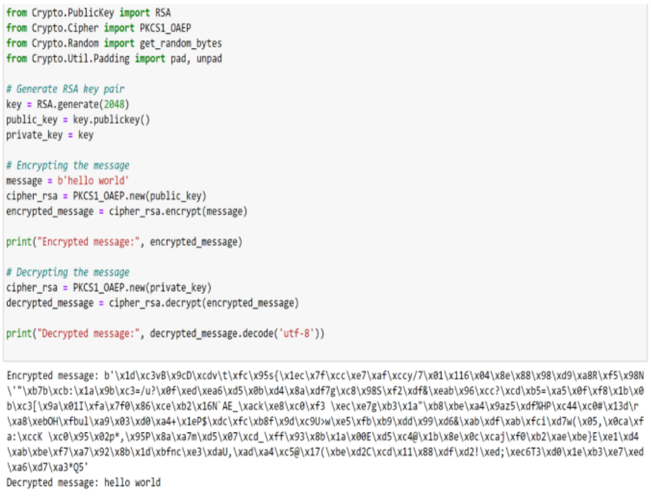
Incorporating heightened levels of security and effectiveness in public key cryptography to combat quantum threats involves a series of procedures, including selecting suitable postquantum algorithms, integrating these algorithms into existing cryptographic frameworks, and evaluating their efficacy and security. This section elaborates on the practical aspects of implementing these post-quantum cryptographic solutions, emphasizing the steps taken to achieve the desired improvements, the challenges encountered, and the results obtained. The implementation was carried out using Jupyter Notebook (Python 3.11.5) on a machine with the following specifications: Intel Core i7 processor, 16 GB RAM, and Windows 11 operating system. The libraries and tools used included RSA 4.9 Python, NumPy, SciPy, Socket, Pycryptodome, ECDSA, PQ Clean (a quantum-safe cryptographic library for Kyber), Wireshark, and RsaCtfTool. During the implementation phase, cryptographic algorithms were developed and executed using Python. The RSA algorithm was implemented using Python's built-in libraries, focusing on key generation, encryption, and decryption processes. The Kyber post-quantum cryptographic algorithm was implemented using PQ Clean, which included key encapsulation mechanisms (KEM) and public key encryption (PKE) schemes. Furthermore, a hybrid cryptographic scheme combining RSA and Kyber was devised to leverage the strengths of both algorithms. This hybrid scheme involved encrypting data with RSA and Kyber KEM, aiming to assess potential improvements in security and performance.

A. *Selection and implementation of classical algorithms(PKC)*

Based on NIST's recommendations, we selected the following algorithms for our implementation [21]: RSA: Widely used for secure data transmission, RSA provides robust public key encryption and digital signatures based on the computational difficulty of factoring large integers. NIST recommends RSA for its well-established security and widespread adoption in various cryptographic applications.

RSA Python code implementation steps. Step 1. Laboratory preparation, which entails setting up our Python in Jupyter Notebook. Step 2. Importation of Pycryptodome libraries. Step 3. Generating the RSA public and private key using the imported library. Step 4. Encrypting the message (hello world), which serves as our data with the public key. Step 5. Printing out the encrypted message. Step 6. Decrypting the message with the private key. Step 7. Printing out the decrypted message – Hello World

Figure 1 illustrates the entire process and the RSA Python code implementation outputs.



```python
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
from Crypto.Random import get_random_bytes
from Crypto.Util.Padding import pad, unpad

# Generate RSA key pair
key = RSA.generate(2048)
public_key = key.publickey()
private_key = key

# Encrypting the message
message = b'hello world'
cipher_rsa = PKCS1_OAEP.new(public_key)
encrypted_message = cipher_rsa.encrypt(message)

print("Encrypted message:", encrypted_message)

# Decrypting the message
cipher_rsa = PKCS1_OAEP.new(private_key)
decrypted_message = cipher_rsa.decrypt(encrypted_message)

print("Decrypted message:", decrypted_message.decode('utf-8'))
```

```
Encrypted message: b'\x1d\xc3vB\x9cD\xcdv\t\xfc\x95s{\x1ec\x7f\xcc\xe7\xaf\xccy/7\x01\x116\x04\x8e\x88\x98\xd9\xa8R\xf5\x98N
\'"\xb7b\xcb:\x1a\x9b\xc3=/u?\x0f\xed\xea6\xd5\x0b\xd4\x8a\xdf7g\xc8\x98S\xf2\xdf&\xeab\x96\xcc?\xcd\xb5=\xa5\x0f\xf8\x1b\x0
b\xc3[\x9a\x01I\xfa\x7f0\x86\xce\xb2\x16N`AE_\xack\xe8\xc0\xf3 \xec\xe7g\xb3\x1a"\xb8\xbe\xa4\x9az5\xdf%HP\xc44\xc0#\x13d\r
\xa8\xebOH\xfbul\xa9\x03\xd0\xa4+\x1eP$\xdc\xfc\xb8f\x9d\xc9U>w\xe5\xfb\xb9\xdd\x99\xd6&\xab\xdf\xab\xfci\xd7w(\x05,\x0ca\xf
a:\xccK \xc0\x95\x02p*,\x95P\x8a\xa7m\xd5\x07\xcd_\xff\x93\x8b\x1a\x00E\xd5\xc4@\x1b\x8e\x0c\xcaj\xf0\xb2\xae\xbe}E\xe1\xd4
\xab\xbe\xf7\xa7\x92\x8b\x1d\xbfnc\xe3\xdaU,\xad\xa4\xc5@\x17(\xbe\xd2C\xcd\x11\x88\xdf\xd2!\xed;\xec6T3\xd0\x1e\xb3\xe7\xed
\xa6\xd7\xa3*Q5'
Decrypted message: hello world
```

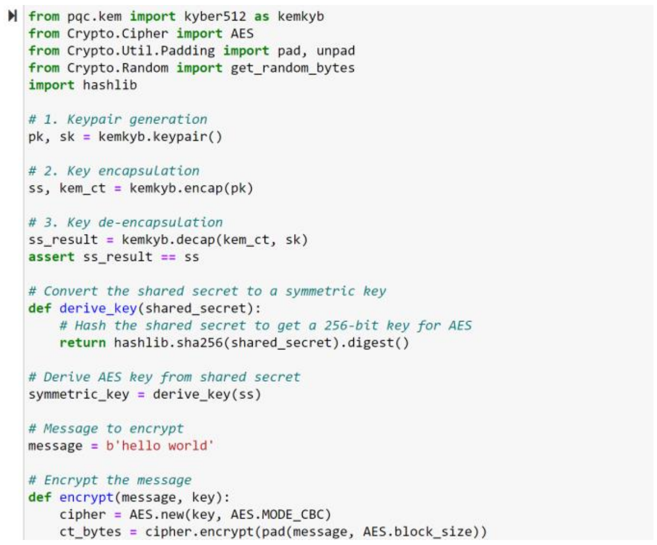
Fig 1. RSA Python Code Implementation

B. *Implementation of Post Quantum Algorithms*

Based on NIST's recommendations, we selected the following algorithms for our implementation [21]: Crystals-Kyber for crucial encapsulation.

The algorithm was chosen due to its strong security guarantees against quantum attacks and efficient practical application performance. The selection was based on the availability of the Python library for implementation. Crystal-Kyber 512 Python code implementation steps. Step 1. Laboratory preparation, which entails setting up our Python in Jupyter Notebook. Step 2. Importation of PQ Clean libraries. Step 3. Generating the secret and public key pair with the imported library. Step 4. Generates symmetric key (AES) and encrypts with public key, which produces a cipher text. Step 5. The message (hello world) is encrypted along with the ciphertext. Step 6. Encrypted message and ciphertext printed out—step 7. Message and ciphertext decapsulated with private key. Step 8. Decrypted message printed out. The screenshot below illustrates the entire process along with the outputs.

Figure 2 depicts the Implementation of Post-Quantum Algorithms and processes used for the digital signatures, both for the classical and post-quantum algorithms: ECDSA for classical and SPHICS+ for PQC.



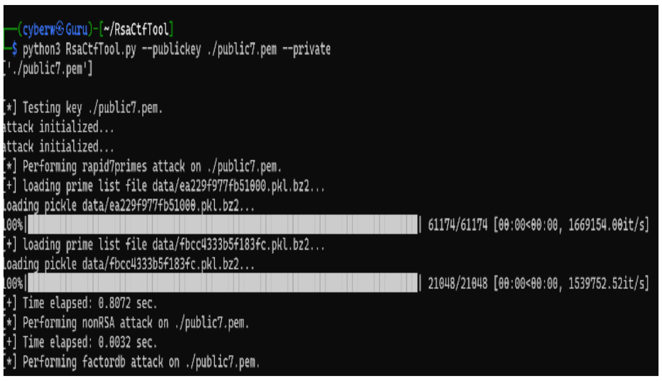
```python
from pqc.kem import kyber512 as kemkyb
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes
import hashlib

# 1. Keypair generation
pk, sk = kemkyb.keypair()

# 2. Key encapsulation
ss, kem_ct = kemkyb.encap(pk)

# 3. Key de-encapsulation
ss_result = kemkyb.decap(kem_ct, sk)
assert ss_result == ss

# Convert the shared secret to a symmetric key
def derive_key(shared_secret):
    # Hash the shared secret to get a 256-bit key for AES
    return hashlib.sha256(shared_secret).digest()

# Derive AES key from shared secret
symmetric_key = derive_key(ss)

# Message to encrypt
message = b'hello world'

# Encrypt the message
def encrypt(message, key):
    cipher = AES.new(key, AES.MODE_CBC)
    ct_bytes = cipher.encrypt(pad(message, AES.block_size))
```

Fig 2. Implementation of Post Quantum Algorithms

C. *Cracking weak RSA Keys;*

A Python algorithm with a cracking tool (RsaCtfTool) was implemented in Kali Linux to crack weak RSA keys. Weak RSA keys were generated with an imported RSA Python library, as weak RSA keys are prevented by default from being used with most popular libraries. The algorithm was tested on RSA keys with various bit lengths to evaluate its effectiveness.

The same tool was applied to KEM encrypted data with no success. The table below shows the comparative analysis between RSA and KEM implementations.



```
─(cyberw㉿Guru)-[~/RsaCtfTool]
─$ python3 RsaCtfTool.py --publickey ./public7.pem --private
['./public7.pem']

[*] Testing key ./public7.pem.
ttack initialized...
ttack initialized...
[*] Performing rapid7primes attack on ./public7.pem.
[+] loading prime list file data/ea229f977fb51000.pkl.bz2...
loading pickle data/ea229f977fb51000.pkl.bz2...
00%|████████████████████████████████████████| 61174/61174 [00:00<00:00, 1669154.00it/s]
[+] loading prime list file data/fbcc4333b5f183fc.pkl.bz2...
loading pickle data/fbcc4333b5f183fc.pkl.bz2...
00%|████████████████████████████████████████| 21048/21048 [00:00<00:00, 1539752.52it/s]
[+] Time elapsed: 0.8072 sec.
[*] Performing nonRSA attack on ./public7.pem.
[+] Time elapsed: 0.0032 sec.
[*] Performing factordb attack on ./public7.pem.
```

Fig 3. RsaCtfTool -RSA cracking implementation.

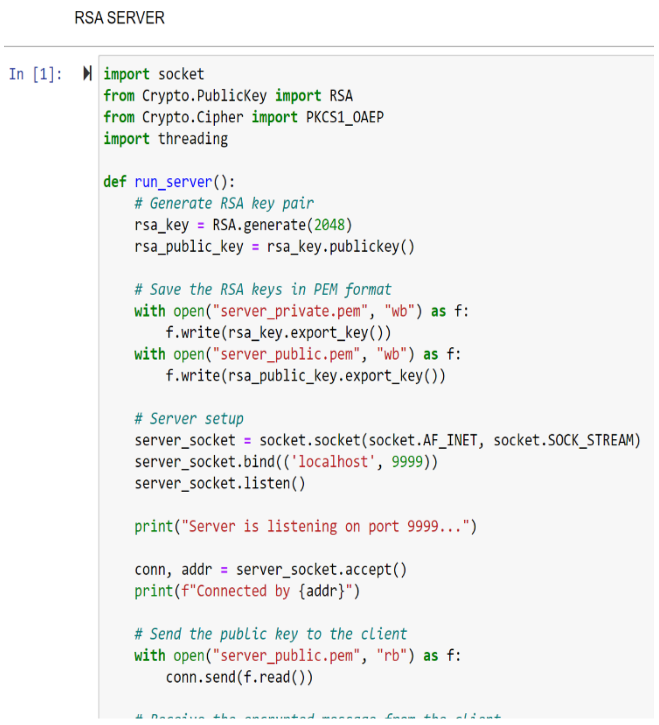
TABLE 1. SECURITY COMPARATIVE ANALYSIS OF RSA AND KYBER

Aspect	RSA Encryption	KEM Encryption
Cryptographic Basis.	Relies on the difficulty of factoring large numbers	Based on post-quantum hardness assumptions
Vulnerable to Classical Attacks.	Yes, susceptible to various attacks (e.g., low public exponent, weak primes)	No, immune to these classical attacks. Demonstrate with RsaCtfTool: cracks RSA; fails on KEM.
RsaCtfTool Attack Feasibility	Vulnerable to RsaCtfTool attacks, including factorization and Wiener's attack.	Not susceptible, due to fundamentally different structure.
Resistance to Factorization	Vulnerable depends on factorization difficulty.	Resistant, does not use number-theoretic assumptions like factorization.
Quantum Resistance.	Not quantum-resistant, vulnerable to Shor's algorithm.	Designed to resist quantum attacks (e.g., Kyber, using lattice-based problems).
Security in Practice.	Secure when configured with large keys and correct settings, but weak to misconfigurations.	Designed to maintain security against classical and quantum attacks, even with minor misconfigurations.
Implementation Complexity	Simple and widely understood but can suffer from configuration issues.	More complex, especially for post-quantum schemes, yet highly secure against typical vulnerabilities.
Known Attacks	Prone to factorization attacks, Wiener's, and fault attacks if poorly implemented.	KEM:** Resistant to these attacks, protected by different cryptographic assumptions.

D. *RSA and KEM Server to Client Transmission*

a) Implementation: A primary server-to-client transmission was set up using Python libraries. This set was implemented with both RSA and KEM cryptographic libraries separately, and a comparative analysis was performed using Wireshark. The following steps were followed to set up the RSA server-to-client transmission.

- Step 1. Laboratory preparation, which entails setting up our Python in Jupyter Notebook.
- Step 2. Importation of Pycryptodome and Socket libraries.
- Step 3. Generating the secret and public key pair with the imported library.
- Step 4. Saving RSA keys in PEM format.
- Step 5. Configuring server setup with Socket library. A server configured to use a localhost IP address – 127.0.0.1 and port number 9999.
- Step 6. The server output is configured to print the output ('Server is listening on port 9999').
- Step 7. A server is configured to receive incoming connections on the above port and to print out ('server connected') once the incoming connection is received on port 9999.
- Step 8. Server sends public key to the client.
- Step 9. Server receives encrypted message from the client.
- Step 10. Server decrypts the message with its private key.
- Step 11. The server prints out the decrypted message.
- Step 12. The server closes the connection once the transmission is completed. Transmission was completed upon receiving an encrypted message from the client.


RSA SERVER

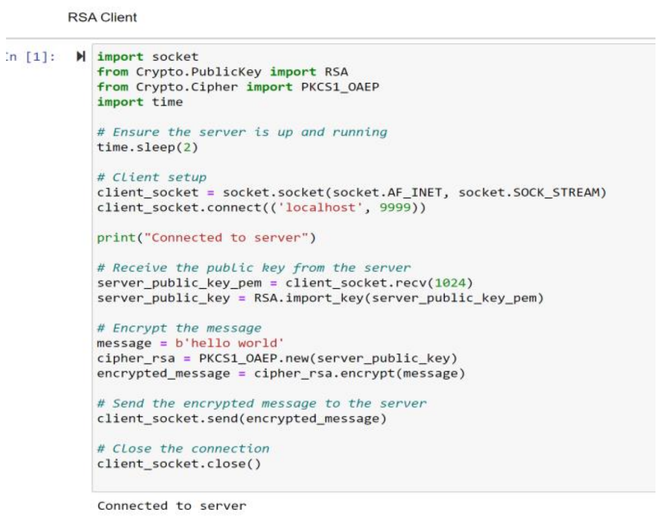
```
In [1]:  import socket
         from Crypto.PublicKey import RSA
         from Crypto.Cipher import PKCS1_OAEP
         import threading

         def run_server():
             # Generate RSA key pair
             rsa_key = RSA.generate(2048)
             rsa_public_key = rsa_key.publickey()

             # Save the RSA keys in PEM format
             with open("server_private.pem", "wb") as f:
                 f.write(rsa_key.export_key())
             with open("server_public.pem", "wb") as f:
                 f.write(rsa_public_key.export_key())

             # Server setup
             server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
             server_socket.bind(('localhost', 9999))
             server_socket.listen()

             print("Server is listening on port 9999...")

             conn, addr = server_socket.accept()
             print(f"Connected by {addr}")

             # Send the public key to the client
             with open("server_public.pem", "rb") as f:
                 conn.send(f.read())
```

Fig 4. RSA and KEM Server to server-to-client transmission

Figure 4 depicts the client's Python implementation of RSA, with the output showing the connection to the server in secure transmission. The following steps were followed to set up the RSA client-to-server transmission.

- Step 1. Configuring client port to connect to server port address of 9999.
- Step 2. Configuring client output to read (connected to the server) once the connection to port 9999 is made.
- Step 3. The client port receives the public key from the server port, encrypts the message 'hello world', and sends it to the server.
- Step 4. Client closes port after successfully encrypting and transmitting message securely to server.

The server implementation is illustrated in the image below, while the client implementation follows. The implementation was also set up for the Kyber server-to-client transmission. The same steps were implemented for the Kyber server-to-client transmission, with the importation of the PQ clean libraries in place of the Pycryptodome libraries.


RSA Client

```
In [1]:  import socket
         from Crypto.PublicKey import RSA
         from Crypto.Cipher import PKCS1_OAEP
         import time

         # Ensure the server is up and running
         time.sleep(2)

         # Client setup
         client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
         client_socket.connect(('localhost', 9999))

         print("Connected to server")

         # Receive the public key from the server
         server_public_key_pem = client_socket.recv(1024)
         server_public_key = RSA.import_key(server_public_key_pem)

         # Encrypt the message
         message = b'hello world'
         cipher_rsa = PKCS1_OAEP.new(server_public_key)
         encrypted_message = cipher_rsa.encrypt(message)

         # Send the encrypted message to the server
         client_socket.send(encrypted_message)

         # Close the connection
         client_socket.close()

         Connected to server
```

Fig 5. RSA Client Implementation

V. Results and discussions

This section introduces the outcomes of the execution and evaluation of RSA and Kyber algorithms alongside a blended approach that integrates both. Furthermore, it encompasses the outcomes of a simulated interaction between a client and server employing RSA and Kyber, evaluated using Wireshark and an algorithm developed in Python for decrypting vulnerable RSA keys. The analysis of these findings pertains to efficiency, robustness, and real-world ramifications. The performance evaluation was analyzed.

VI. Comparative Analysis

The findings suggest that the RSA algorithm demonstrates slower speed in key generation, encryption, and decryption; the Kyber algorithm and the hybrid approach present a heightened level of security and are much faster, especially within a post-quantum environment. Despite being much faster, the hybrid method integrates the advantages of both RSA and Kyber, thus furnishing a harmonious blend of security and operational effectiveness. The time spent in generating the hybrid and Kyber keys is not even up to a millisecond resulting in the rounding up to zero. The RSA algorithm is slower in decryption as it is more complex for the computation involved in obtaining the secret key.

Table 2. Benchmarking results of RSA and kyber

Algorithm	Key Generation Time	Encryption Time	Decryption Time
CRYSTALS-Kyber	0.001018s	0.0000 s	0.0000 s
RSA-2048	0.911545s	0.0000s	0.016297s

The client-server simulation results show that RSA is slightly faster in key exchange, encryption, and decryption compared to Kyber. This was due to the Kyber simulation also processing the AES symmetric key transmission, while the RSA simulation did not conduct any symmetric key transmission. However, the performance difference is marginal, and Kyber’s post-quantum security makes it a preferable choice for future-proof cryptographic applications.

The results indicate that weak RSA keys (e.g., 32-bit) can be cracked quickly, highlighting the importance of using sufficiently large key sizes to ensure security. The algorithm could not crack 1024-bit keys within a reasonable time limit, demonstrating that larger key sizes provide better security against brute-force attacks. However, these larger key sizes would be inadequate against the sheer force and computing power of quantum computing. We cannot simulate the quantum computing process due to the limited resources. Quantum simulators and access to quantum hardware through cloud computing also exist. Simulation and the algorithms, however, are beyond our resources and time limit.

The investigation conducted in this paper delved into the integration and effectiveness of traditional and post-quantum cryptographic algorithms, specifically focusing on RSA and Kyber, in response to emerging quantum threats. The deployment of these algorithms and their combined approach underwent thorough examination and evaluation. Presented below are the primary discoveries. Evaluation of Performance Metrics: RSA: The time taken for encryption: 0.0000 seconds. The time taken for decryption: 0.016297 seconds. Kyber: Kyber exhibited robust performance, displaying encryption and decryption durations superior to RSA, thus indicating its potential as an alternative in the post-quantum realm. Hybrid Approach: The amalgamated cryptographic strategy, which amalgamates RSA and Kyber, accomplished a harmonious blend of classical and post-quantum security. This strategy guarantees resilience against quantum threats while upholding efficiency in encryption and decryption procedures. Simulation of Client-Server Interaction: Through utilizing Wireshark, the client-server interactions involving RSA and Kyber unveiled that the KEM approach maintained secure communication without notable delays or performance deterioration, thus demonstrating its practical utility. Analysis of Cryptographic Systems: The RsaCtfTool effectively penetrated feeble RSA keys (32-bit and 64-bit), illustrating the susceptibilities of keys with low bit-lengths. This underscores the imperative need for more robust key sizes and the migration towards postquantum algorithms such as Kyber. The examination carried out in the paper offers a thorough assessment of traditional and post-quantum cryptographic algorithms, particularly RSA and Kyber. The findings underscore the significance of embracing post-quantum cryptography to safeguard data integrity in the era of quantum computation. The combined strategy merging RSA and Kyber provides an interim solution, leveraging the capabilities of both algorithms to mitigate current and future cryptographic risks. The performance evaluations and client-server interactions validate the feasibility and efficacy of these cryptographic solutions in practical scenarios. Furthermore, analyzing vulnerable RSA keys emphasizes the pressing need for enhanced cryptographic safeguards.

VII. conclusion

The implementation and benchmarking outcomes highlight the efficacy of both RSA and Kyber algorithms in delivering secure cryptographic remedies. The hybrid scheme introduces a pragmatic strategy for harmonizing performance and security. The client-server simulation and the key-cracking algorithm further accentuate the criticality of judiciously selecting cryptographic algorithms and key dimensions to guarantee robust security in both classical and post-quantum scenarios. The discoveries of this chapter furnish valuable perspectives into the performance and security repercussions of classical and post-quantum cryptographic algorithms, guiding the formulation of secure and effective cryptographic systems for the forthcoming era.

Future works will consider optimizing hybrid schemes for diverse applications, examining adaptive key management in fluctuating environments, and exploring alternative post-quantum cryptographic algorithms to bolster security. Implementing such frameworks in practical systems, alongside assessing their efficacy and security, will serve as pivotal stages in the progression of cryptography.

REFERENCES

[1] Xie, J., Tan, X., & Tan, L., 2022. CR-BA: Public Key Infrastructure Certificate Revocation Scheme Based on Blockchain and Accumulator. *School of Computer Science, Sichuan Normal University, Chengdu, China; Institute of Computer Science, Chinese Academy of Sciences, Beijing, China*. Available at: jkxy_tl@sicnu.edu.cn.

[2] Juniawan, S., 2022. Key Management. In: *Cryptographic Security Concerns on Timestamp Sharing via a Public Channel in Quantum-Key-Distribution Systems*. Springer, pp.105-117. https://doi.org/10.1007/978-1-4842-7486-6_8

[3] Papadopoulos, I. and Wang, J., 2023. Polar Codes for Module-LWE Public Key Encryption: The Case of Kyber. *Cryptography*, 7(1), p.2. https://doi.org/10.3390/cryptography7010002.

[4] Dang, V.B., Mohajerani, K. and Gaj, K., 2023. High-Speed Hardware Architectures and FPGA Benchmarking of CRYSTALS-Kyber, NTRU, and Saber. *IEEE Transactions on Computers*, 72(2), pp.306-320. https://doi.org/10.1109/tc.2022.3222954.

[5] Wang, X., Luo, T. and Li, J., 2018. A More Efficient Fully Homomorphic Encryption Scheme Based on GSW and DM Schemes. *Security and Communication Networks*, 2018, pp.1-14. https://doi.org/10.1155/2018/8706940.

[6] A. Yeboah-Ofori, C. K. Agbodza, F. A. Opoku-Boateng, I. Darvishi and F. Sbai, "Applied Cryptography in Network Systems Security for Cyberattack Prevention," 2021 International Conference on Cyber Security and Internet of Things (ICSIoT), France, 2021, pp. 43-48, doi: 10.1109/ICSIoT55070.2021.00017. keywords: {Fabrication;Hash functions;Public key;Network security;Virtual private networks;Encryption;Cryptography;Applied Cryptography;Network Security;RSA;Interception;Interruption;Modification;Fabrication},

[7] A. Yeboah-Ofori, I. Darvishi and A. S. Opeyemi, "Enhancement of Big Data Security in Cloud Computing Using RSA Algorithm," *2023 10th International Conference on Future Internet of Things and Cloud (FiCloud)*, Marrakesh, Morocco, 2023, pp. 312-319, doi: 10.1109/FiCloud58648.2023.00053.

[8] A. Yeboah-Ofori and A. Ganiyu, "Big Data Security Using RSA Algorithms in A VPN Domain," *2024 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*, Victoria, Seychelles, 2024, pp. 1-6, doi: 10.1109/ACDSA59508.2024.10467364.

[9] Borges, F., Reis, P.R. and Pereira, D., 2020. A Comparison of Security and its Performance for Key Agreements in Post-Quantum Cryptography. *IEEE Access*, 8, pp.142413-142422. https://doi.org/10.1109/ACCESS.2020.3013250.

[10] Wang, X., Luo, T. and Li, J., 2018. A More Efficient Fully Homomorphic Encryption Scheme Based on GSW and DM Schemes. *Security and Communication Networks*, 2018, pp.1-14. https://doi.org/10.1155/2018/8706940.

[11] Koseki, R., Ito, A., Ueno, R., Tibouchi, M. and Homma, N., 2022. Homomorphic encryption for stochastic computing. *Journal of Cryptographic Engineering*, 13(2), pp.251- 263. https://doi.org/10.1007/s13389-022-00299-6.

[12] Verma, S. and Garg, D., 2011. Improvement in RSA cryptosystem. *Journal of Advances in Information Technology*, 2(3), pp.146-151

[13] Shuhab, S., Farina, R., Rabia, R., Sanam, S.R., Shahab, A. and Abdulla, S.A., 2022. An Enhanced Architecture to Resolve Public-Key Cryptographic Issues in the Internet of Things (IoT), Employing Quantum Computing Supremacy. *Sensors*, 22(21), p.8151. https://doi.org/10.3390/s22218151.

[14] Liu-Jun, Wang., You, Zhou., Jianfeng, Yin., Qing, Chen. (2022). Authentication of quantum key distribution with post-quantum cryptography and replay attacks. arXiv.org, Available from: 10.48550/arXiv.2206.01164

[15] Doliskani, J., 2021. Efficient Quantum Public-Key Encryption From Learning With Errors. *arXiv: Quantum Physics*. Available at: https://doi.org/10.48550/arxiv.2103.13788.

[16] Melis, P., Durak, K. and Tefek, U., 2022. Cryptographic security concerns on timestamp sharing via a public channel in quantum-key-distribution systems. *Physical Review A*, 106(1), p.012611. https://doi.org/10.1103/PhysRevA.106.012611

[17] Pereira, M., Kato, G., Curty, M. and Tamaki, K., 2023. A security framework for quantum key distribution implementations. *Electronics*, 12(6), p.2643. https://doi.org/10.3390/electronics12122643

[18] Randy, Kuang., Maria, Perepechaenko., Ryan, Toth., Michel, Barbeau. (2022). Benchmark Performance of a New Quantum-Safe Multivariate Polynomial Digital Signature Algorithm. Available from: 10.1109/QCE53715.2022.00067

[19] Sergey, E., Yunakovsky., Maxim, Kot., N.O., Pozhar., Denis, Nabokov., Mikhail, A., Kudinov., Anton, Guglya., Evgeniy, O., Kiktenko., Ekaterina, Kolycheva., Alexander, Borisov., Aleksey, Fedorov. (2021). Towards security recommendations for public-key infrastructures for production environments in the post-quantum era. EPJ Quantum Technology, Available from: 10.1140/EPJQT/S40507-021-00104-Z

[20] National Institute of Standards and Technology (NIST), 2022. NIST Announces First Four Quantum-Resistant Cryptographic Algorithms. [online] Available at: https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistantcryptographic algorithms [Accessed 4, May 2024]

[21] Barker, E., Barker, W., Burr, W., Polk, W. & Smid, M., 2016. Recommendation for Key Management – Part 1: General (Revision 4). Gaithersburg, MD: National Institute of Standards and Technology.https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf.