Application of Quantum Algorithms for Network Protocols

Vinutna Kolachana, Dolly Upmandewan, Arpit Giri, N. Pavan, Anees Ahmed, M. N. Thippeswamy, and T. R. Vinay

Abstract Cryptographic processes hold immense power in securing data transmissions across the Internet in the modern age. Preparing for future hacking possibilities is deemed essential and crucial to combat data breaches and leaks. With the advent of quantum computing, a field capable of performing complex operations within a short time, breaking into the cryptographic system keys with sheer brute force is visibly possible. This paper aims to create a framework that runs post-quantum cryptographic algorithms shortlisted by NIST for Round 3 on the TLS protocol. The framework is built using Python with the help of LibOQS packages, programmed to work as an API, and invoked with the help of a web application. It allows cross-platform execution as it is contained in a Docker container. The established framework utilizes digital signature cryptography to verify the authenticity of a signed message and key encapsulation mechanism (KEM) for secure communication between client and server. The algorithms can be integrated with various internet-based applications like IoT, blockchain, but the results are demonstrated using a web application for this research.

Keywords Quantum computing · Cryptography · Post-quantum cryptography · TLS · Digital signature · Key encapsulation mechanism · Network protocols

V. Kolachana (✉) · D. Upmandewan · A. Giri · N. Pavan · M. N. Thippeswamy · T. R. Vinay
Department of Computer Science and Engineering, NMIT, Yelahanka, Bangalore 560064, India

M. N. Thippeswamy
e-mail: mntswamy@msrit.edu

T. R. Vinay
e-mail: vinay.tr@nmit.ac.in

A. Ahmed
Unisys India Private Limited, Bangalore, India
e-mail: anees.ahmed@in.unisys.com

V. Bindhu et al. (eds.), *Proceedings of Third International Conference on Communication, Computing and Electronics Systems*, Lecture Notes in Electrical Engineering 844,
https://doi.org/10.1007/978-981-16-8862-1_29

Introduction

Quantum computing is a field of science that deals with quantum mechanics concepts and their application in computing, such that solutions to problems requiring massive computation and operations are made faster and simpler for execution, having lesser energy consumption. The purpose is to develop computer technology based on the behavior of energy and matter at an atomic or subatomic level. Quantum computing could contribute greatly in the fields of finance, military affairs, intelligence, drug design and discovery, aerospace designing, utilities (nuclear fusion), polymer design, AI and big data search, and digital manufacturing [1].

Quantum computing, in short, is the amalgamation of computer science and quantum mechanics, where quantum physical properties of subatomic particles are explored and integrated into computer systems. The most important quantum mechanics principles utilized in this field are superposition, interference, and entanglement. Superposition is the principle wherein a quantum system can exist in multiple quantum states at the same time, until measured at one particular instance, to find its exact state. It is a state where an entity with different measures of a common attribute exhibits all measures simultaneously, making the result a combination of all individual measures [2]. Interference is an extension such that two quantum systems, when superposed, cause the result to either be a sum or difference of their individual measures. Interference can help control the output to lean toward the measure that the user prefers [3]. Quantum entanglement is another property wherein two quantum states are entangled such that their states are not determinable but are correlated. This means that both states are dependent on each other, though they may be in different planes at that instance [4].

In ordinary computers, the simplest functional unit is a "bit" that can have a value of either 1 or 0. In quantum computers, the simplest unit is a "qubit", with the property of having a value of both 1 and 0 at the same time. This is due to the concept of superposition, which allows the qubit to be in two states simultaneously. The rule of thumb is that the total probability of the qubit having state 0 and state 1 must be 100% at all times. The superposition principle allows qubits to perform computations at exponential rates, thereby contributing to the success of quantum computers as we know them [5].

The advent of this new form of computing has brought its share of pros and cons. The advantage is a wide range of enhanced computations in the fields mentioned above. However, the disadvantage posed is that there would be no control once the computation is made mainstream. With the sheer capacity of the system, it would be very easy for the computer to crack tricky passwords and cryptographic keys even by using a brute force method, which is deemed impossible for classical computers. All privacy measures taken now would be utterly useless once the quantum revolution picks its pace.

Dr. Krysta Svore of Microsoft Research stated—"The RSA-2048 Problem would take 1 Billion Years with a Classical Computer, but a Quantum Computer could do it in 100 s!". That is the exponential power of the systems. To highlight the

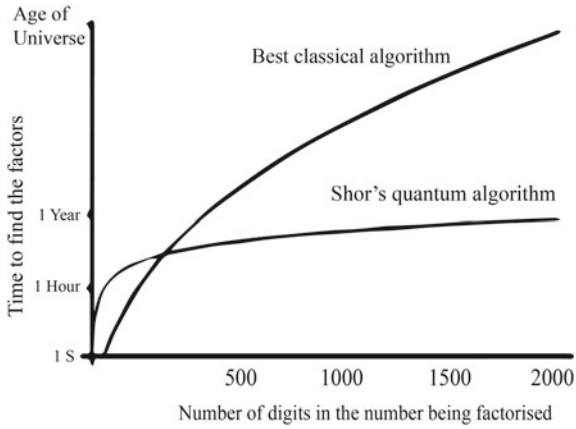


Fig. 1 Comparison of Shor's quantum algorithm with the best classical algorithm

computational efficiency of quantum computers, Fig. 1 shows how much faster Shor's algorithm can decode the prime factor keys of cryptographic algorithms [6].

This brought out a new branch in quantum computing called Post-Quantum Cryptography which focuses on the security of both classical and quantum computers, assuming the wake of a quantum world in the future. It deals with devising encryption or network security algorithms such that both classical and quantum computers are unable to break through the security code. This is required for the current generation when quantum computers are on the rise. It is necessary to build the network's security to avoid breaches of data and information [7]. This field is still relatively unexplored and larger organizations like the National Institute of Standards and Technology (NIST) are researching methods to enhance security. Standardization of such privacy methods would be a breakthrough in the field of computer science and networking.

The novel feature of this research work is that the developed framework utilizes quantum-safe algorithms for cryptography over the Transport Layer Security (TLS) network protocol to encrypt the communication holistically. Not only does the framework differ due to the application of quantum-safe algorithms, but also because it utilizes all of the 7 NIST shortlisted algorithms for encryption, not just one. The framework applies a user-friendly approach as well, by allowing the user to choose what algorithm needs to be applied. An important note is that quantum-safe algorithms do not work the same way as traditional cryptographic algorithms like AES and RSA. They utilize mathematical properties that are different from prime factorization, thereby making key identification by brute force an extremely hard task.

Literature Review

Quantum computing is a new way of computing that is based on quantum mechanics and its unique properties. The studied literature includes the basics of quantum computing, quantum cryptography, network protocols, NIST shortlisted algorithms, identifying the limitations of current cryptographic methods, and many other supportive resources related to the topic.

Firstly, it is necessary to understand the principles, concepts, mathematics, and the working of quantum computers and how they differ from the way classical computers work. To get a comprehensive view of the field, the works of Aaronson of [8] and Rieffel and Polak of [9] prove worthwhile. Considered as the best textbooks by the pioneers of the field, the two sources provide extensive material on the ideologies of quantum computing. Another immaculate source is the book by Sutor [10] which emphasizes the need for quantum computing and how quantum computers work, focusing spectacularly on the computation part of the field rather than just the mathematics.

To understand the need for quantum cryptography, Grau, the author of [11] deems it necessary to know how efficient quantum computers are and how the encryption can be easily broken using algorithms like Shor's algorithm. Mosca's inequality also determines that the encryption algorithm must be ready before quantum computers are commercially available. Even Hoursanov of [12] mentions that quantum-safe algorithms and their implementations must be developed beforehand as development cannot be done instantaneously and getting all protocols to function correctly takes a lot of time.

The authors of [13] give a thorough insight on different cryptographic types like X.509 certificates, IKEv2, SSH, and S/MIME to identify the loopholes of current cryptography. The authors of [14] too describe the current systems in detail and point to the areas that can be infiltrated.

There are a few open-quantum algorithms currently being explored. Lyubashevsky et al. [15] explore lattice-based cryptography to construct security primitives. Quantum-Safe Security Position Paper [16] and Quantum Security Technologies [17] speak of quantum key distribution which is based on quantum physics and allows keys to be exchanged between different locations using the quantum properties of quantum computers. As far as network protocols are concerned, Easterbrook et al. [18] are researching post-quantum TLS, a set of protocols included under transport layer security that limits the vulnerability of key exchange and authentication. Sikeridis et al. [19] speak about the performance of post-quantum algorithms on TLS 1.3 and Sikeridis et al. [20] assess the overhead of post-quantum cryptography in TLS 1.3 and SSH. Crockett et al. [21] attempt to adapt the existing TLS and SSH to incorporate the algorithms.

Lastly, Alagic et al. [22] thoroughly studied to analyze the algorithms that have to be implemented for this work. The entire research runs on the shortlisting done by NIST, its categories, and findings, so this source is of immense importance. NIST has conducted two rounds of shortlisting prior to this research and has arrived at

15 algorithms (7 finalists and 8 alternate selections) among hundreds of entries. The competition is still underway and is expected to conclude in 2023 when a new standard of post-quantum cryptography would be declared for commercial use.

The biggest source of information for this research work is the Open Quantum-Safe (OQS) initiative. Under this collaborative project, there are two segments—LibOQS and protocols/applications. Under the LibOQS scheme, a C library for NIST shortlisted post-quantum algorithms has been developed by quantum computing experts with sponsorship from companies like Microsoft and IBM. The protocols/applications section has certain features that can be incorporated for protocols like TLS, SSH, and X.509 [23]. The Python versions of these algorithms have been utilized for the research.

Brief Overview of the Technologies Used

Quantum Key Distribution (QKD)

Quantum Key Distribution (QKD), also informally called quantum cryptography, uses quantum particles to provide secure symmetric key distribution among the communicating entities. The process of QKD is such that the underlying protocol being used would require to transmit the public key or symmetric key to the receiver of the message for decryption. But this transfer, by itself, can experience eavesdropping and can then threaten the message communication. QKD quantizes this communication such that even if the key transfer is looked upon or is intercepted, the quantum property alters itself and alerts the communicators that the information has been breached. The QKD protocols have been designed to ensure that any eavesdropper in the communication would change the original information being shared, thereby signaling the presence of a third party to the ones communicating with the error being generated. Since qubits are usually made of photons, the quantum properties of light are altered on the interception.

The keys are transmitted among the communicators securely but QKD by itself cannot encrypt the messages as well. Hence, this technology would need to be combined with a conventional symmetric cryptographic encryption like AES to provide end-to-end encryption for messages and make the entire system quantum-safe [24].

Transport Layer Security (TLS)

TLS is the forerunner of Secure Sockets Layer (SSL), and it secures communications over a computer network. This protocol is useful in a variety of applications including email, instant messaging, and voice-over IP.

TLS 1.3, built by Netscape Communications, is the latest version in use to avoid eavesdropping and tampering. It facilitates privacy and data security over the Internet. TLS works mostly on the transport layer which is why it works great at encrypting data from Web sites and applications. To utilize TLS, the system must have a TLS certificate that specifies the domain owner and server's public key and must be issued by a valid certificate authority. The connection between end-users is done through TLS handshaking, a process that incorporates requests and acknowledgements to ensure all data is communicated correctly [25].

OpenSSL

OpenSSL is a robust open-source toolkit for TLS protocol, formerly called the Secure Sockets Layer (SSL). The protocol is built on a full-featured, general-purpose cryptographic library. The SSLeay library, created by Tim J. Hudson and Eric A. Young, is the ancestor of OpenSSL.

The SSL and TLS protocols are implemented in OpenSSL, which is another open-source library. The core C language library implements basic cryptographic functions and provides a variety of utility functions. Wrappers for OpenSSL are available in many programming languages [26].

NIST Shortlisted Algorithms

Anticipating the advent of quantum computers, NIST envisioned a worldwide standard for post-quantum cryptography and opened a global competition for developers to pitch in their quantum-safe algorithms. As per the second round of shortlisting in July 2020, 7 finalists were chosen and 8 were taken as alternative choices. The finale is estimated to be held in 2023. The algorithms all use mathematical domains that differ from traditional prime factorization and can be categorized into five main families, namely code-based, isogeny-based, hash-based, lattice-based, and multivariate system-based [22].

LibOQS, an initiative by the big shots of IT, put together an open-source library containing the codified versions of these algorithms and has paved the way for developers working in the field of post-quantum cryptography to utilize these algorithms effectively by including them in their library. The following two schemes have been mainly used in the research study

Key Encapsulation Scheme—Classic McEliece, NTRU, Kyber, and Saber.

Signature Scheme—Dilithium, Rainbow, and Falcon.

Each algorithm has its own patented technology for creating key pairs and for verifying the corresponding encryption for the key used. Each algorithm uses a certain number of bits for its key pair generation, but the keys that are ultimately utilized for encryption and decryption are 32 bits long.

Table 1 Overview of the NIST algorithm finalists

Scheme	KEM/Signature	Family
Classic McEliece	Key encapsulation	Code-based
Kyber	Key encapsulation	Lattice-based
NTRU	Key encapsulation	Lattice-based
Saber	Key encapsulation	Lattice-based
Dilithium	Digital signature	Lattice-based
Falcon	Digital signature	Lattice-based
Rainbow	Digital signature	Multivariate-based

Table 1 gives an overview of the algorithms selected as the finalists along with their scheme and their family.

LibOQS

LibOQS is a quantum-safe cryptographic library written in C which is open source. LibOQS provides a series of open-source Key Encapsulation Mechanisms (KEM) and Digital Signature algorithms that are quantum-safe.

A common scheme for these algorithms, as well as the test harness and benchmarking routines, are available in the library. Douglas Stebila and Michele Mosca-led Open Quantum-Safe (OQS) initiative aims to build and incorporate quantum-safe cryptography into applications to make implementation easier for real-world problems. Via OpenSSL and OpenSSH, TLS and SSH integrations with LibOQS are established [23].

There is also a Python language wrapper for the library called LibOQS-Python. This allows users to import a specialized package in Python to utilize the classes and functions defined for each of the shortlisted algorithms. It is a derivation from the C library and provides all the essential parameters for the algorithms' usage [27].

Dockerization

Docker is the software for building portable, lightweight containers that help make application development, testing, and deployment easier. Containerization is a form of operating system virtualization that enables users to run multiple applications in isolated containers, all sharing the same operating system. Docker has been used to make sure that no matter what operating system the client is on, data transfer between client and server is always safe from attack by a quantum computer. This has made the system very portable, flexible, and easy to use [28].

Methodology

As specified earlier, the purpose of the research work is to utilize the shortlisted NIST algorithms as they have been defined using LibOQS. This means that the structure and usage of the algorithm are fixed, but the methodology of accessing the algorithm has been defined in this work. Since TLS is being used, the flow has been configured accordingly.

A user's device is considered as the client that is communicating with the server. After applying quantum-safe algorithms for encryption, the client sends the message using the TLS protocol through an HTTPS channel. This channel would ensure security against hackers, quantum/classical computers, and other agencies. The data then goes to the server which performs the decryption and retrieves the message. Figure 2 shows the process in a brief manner. This is a general process that depicts the scope of cryptography.

The architecture of the framework highlighted in Fig. 3 demonstrates the establishment of the environment. The KEM and digital signature algorithms are accessed through the LibOQS-Python package, constituting one segment of the architecture. The other part is that of OpenSSL and TLS wherein the two services are utilized to set up secure communication. A fork of OpenSSL called OQS_OpenSSL_1_1_1 is used to integrate TLS with the quantum-safe algorithms using the certificate and keys generated by them [29]. This entire setup containing the three segments is then enclosed within a container (here, Docker) to run the framework cross-platform. The environment is native to this research work and has been developed solely to facilitate the usage of quantum-safe algorithms.

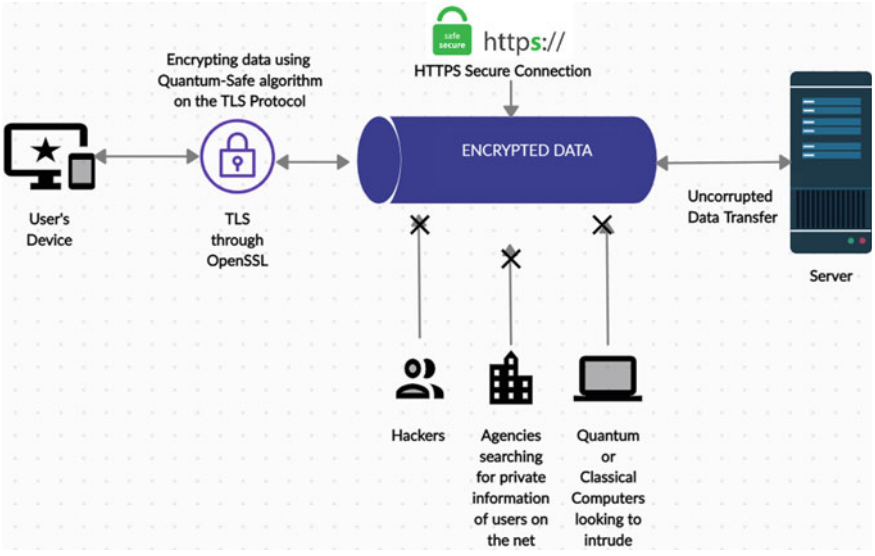


Fig. 2 The design of the system

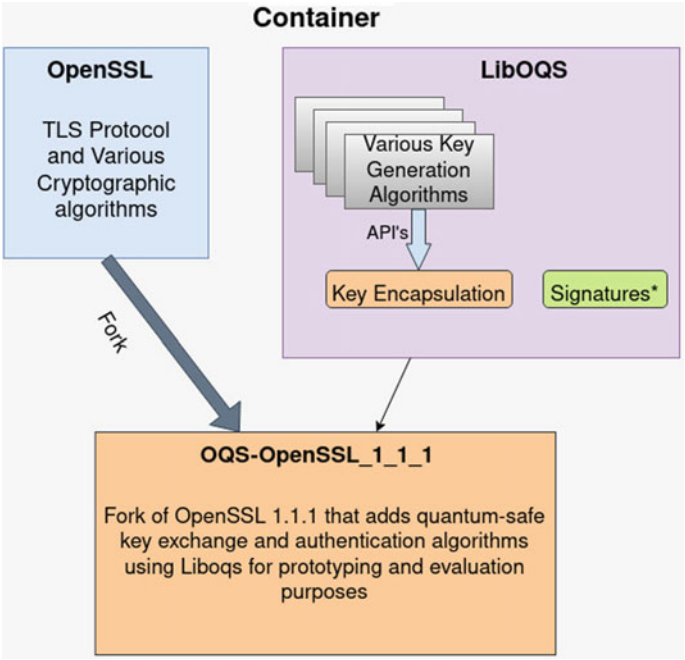


Fig. 3 Architecture of the framework

Additionally, NIST-shortlisted algorithms are divided into two schemes—key encapsulation mechanism and digital signature. It is to be noted that each scheme has its own way of encryption and decryption and that a separate design would be required for each. This means that a separate methodology would be required for the individual schemes as well. Under the following sections, the process of each scheme is elaborated and the approach has been illustrated.

Key Encapsulation Mechanism (KEM)

The Key Encapsulation Mechanism (KEM) scheme is a unique way of communication that deals with two segments—one, the message being sent is to be encrypted/decrypted and two, the keys being used for the message have to be encapsulated/decapsulated. If traditional public-key encryption schemes are to be considered, party A generates the public/private key pair and encrypts its message using the private key. It shares the public key along with the ciphertext to party B, who uses the key to decrypt the ciphertext and retrieve the original message. The drawbacks of this scheme are that the message length may be limited, encryption may not be completely secure, and the keys' mathematics only depend on prime factorization.

To combat the limitations, a scheme called KEM is constructed, containing two layers of communication:

1. A public key layer that establishes a random symmetric key.
2. A symmetric key layer that works to protect the data using the symmetric key generated by the previous layer.

Both these layers work independently to a great extent, thereby assuring the system of greater security. Since there are two forms of keys used to protect the data, even if one set gets compromised, the communicators get notified of the intrusion and appropriate measures are taken [30]. For this research, Advanced Encryption Standard (AES) is being used in the public key layer and the NIST-shortlisted KEM algorithms are used in the symmetric key layer depending on the choice of the user.

For this work, the process mentioned above was tweaked according to the environment established. Certain modifications were made to ensure the end-to-end security of the communication. To illustrate the same, the communication's two endpoints are considered as the client and the server, where the client could be any user's device. The procedure for the communication is as follows:

1. The client would generate the quantum-safe key pair as per the chosen algorithm where the key pair refers to the public key and the private key to be used for communication.
2. The public key is then sent to the server in a secured manner.
3. The server then uses the public key as the input for an encapsulation function that gives an output of a shared secret key and a ciphertext.
4. The server keeps the shared secret key with itself and sends the ciphertext to the client.
5. On receiving the ciphertext, the client uses it as an input for a decapsulation function that produces another secret shared key. The validation is that both the client and the server must match the shared secret keys for the communication to have been intruder-free.
6. Now that it has the shared secret key, the client encrypts the message using the shared secret key for the AES encryption function. It sends the resultant encrypted text to the server.
7. The server finally uses its version of the shared secret key to decrypt the encrypted text and retrieve the original message of the client.

This entire procedure is highlighted in Fig. 4 containing the sequence diagram for the KEM scheme. Figure 5 reflects the corresponding architecture diagram of the scheme. It must be noted that though the communication can be reversed in a real-world situation, this research work contains only the procedure mentioned above.

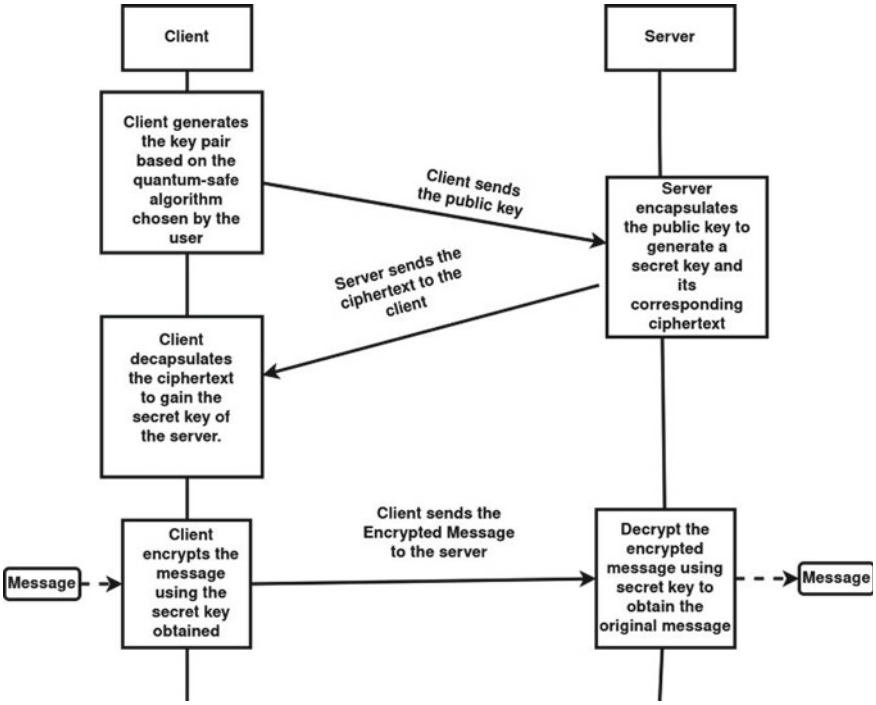


Fig. 4 Sequence diagram of KEM implementation

Digital Signature

Digital signature is a system used to authenticate or verify a message or a document. It works the same way as an e-signature and its purpose is to protect against tampering, deception, and counterfeits. It works on the principle of asymmetric cryptography where two keys, public and private, are used for encryption.

The one who signs the message is referred to as the signer and the other party is referred to as the verifier. The signer signs the intended message using the private key and shares the ciphertext and the public key with the verifier. The verifier then uses it to retrieve the message and then compares it with the original message to see if the signature is authentic. If the verifier finds that the messages are not the same, it means that the message has been meddled with [31].

In this framework, the methodology for digital signature has two endpoints referred to as signer and verifier. The process of digital signature is as follows:

1. The signer generates a key pair based on the chosen quantum-safe algorithm. It generates both a public key and a private key.
2. The signer uses the private key to sign the message and sends the signature to the verifier along with the public key, the selected algorithm, and the original

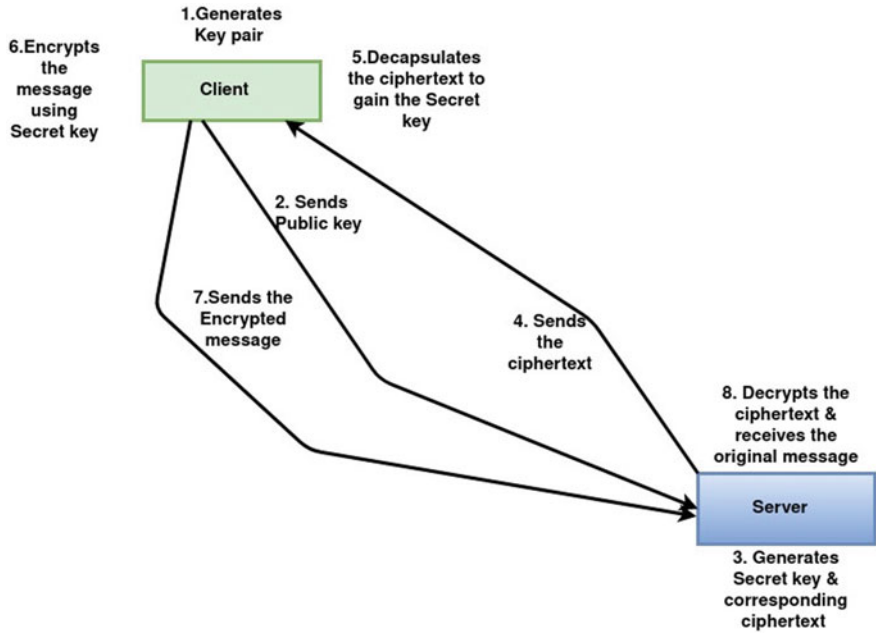


Fig. 5 Architecture diagram of KEM implementation

message for verification. The verifier then retrieves the original message from the signed one using the public key.

3. The verifier compares the original message with the decrypted one to see if they are the same. If they are the same, the signature is accepted as a valid one. Else, the communication has been breached.

The entire sequence has been depicted in Fig. 6 which highlights the process flow. Figure 7 depicts the architecture of the digital signature scheme used in the framework.

Implementation

There are two aspects to establishing the framework—to develop an API with working quantum-safe algorithms and to establish a web application to function alongside the API using TLS 1.3 protocol. The framework is meant to encourage the use of the algorithms in all fields that utilize TLS protocol in any form, like in web applications, cloud platforms, IoT, blockchain, etc. A web application is developed using Flask to demonstrate the usage of the proposed framework.

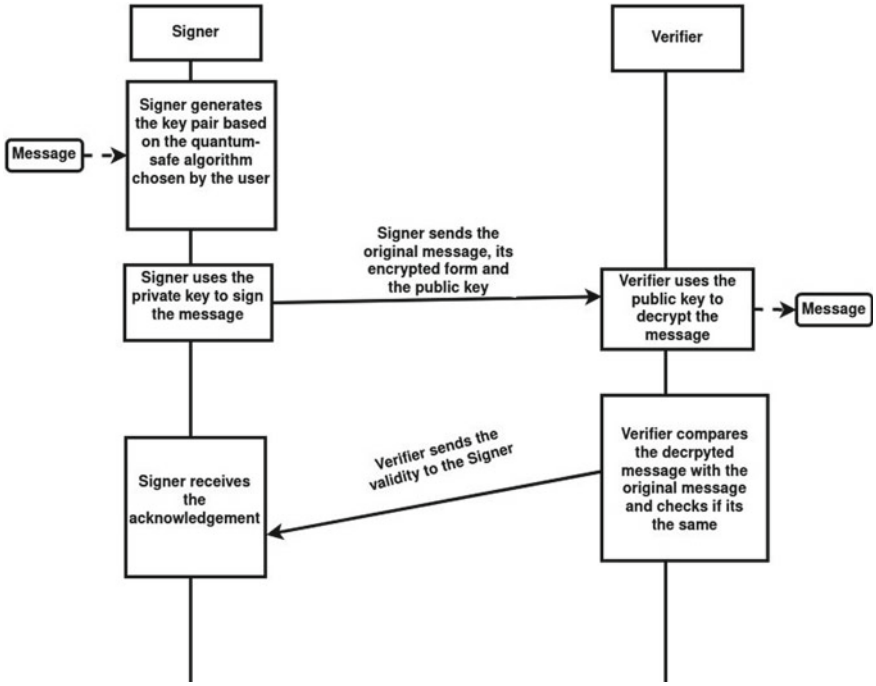


Fig. 6 Sequence diagram of digital signature scheme

API Development

The API includes Python versions of the quantum-safe algorithms and starts with the server–client establishment. In this work, three Open Quantum-Safe libraries have been installed to enable the algorithm functions:

- The OQS_OpenSSL_1_1_1 fork is installed to enable TLS-based certificates and keys that are used to establish an HTTPS connection over TLS 1.3 [29].
- The LibOQS library is installed to get the basic C files of the algorithms along with their mathematical key generation methods [32].
- The LibOQS-Python library is then installed to obtain the Python functions of the algorithms that interact with the LibOQS C library to generate the keys [27].

The LibOQS-Python library provides a Python package called "oqs", which gives the following functions classes and methods on integration:

- Class Signature—To enable the digital signature algorithms and their functions
  - Sigalg variable—A variable containing the algorithm chosen among "DEFAULT" (A default algorithm in case of no choice), "Dilithium2" (Crystals Dilithium algorithm), "Falcon-512" (Falcon algorithm), and "Rainbow-I-Classic" (Rainbow algorithm).

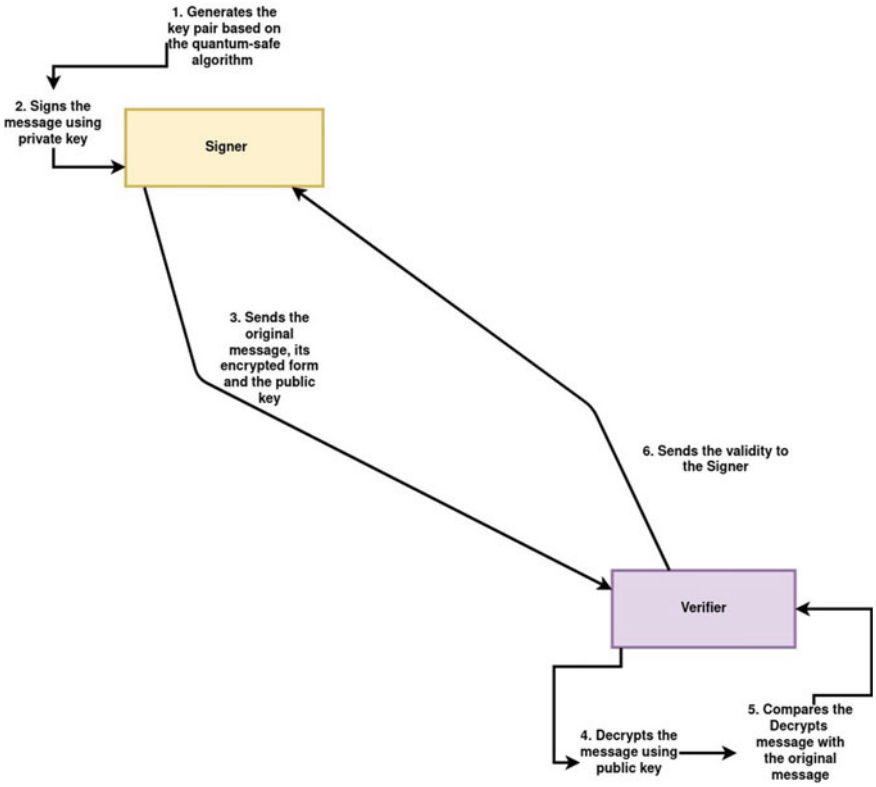


Fig. 7 Architecture diagram of digital signature scheme

- details()—A function that displays the details of the algorithm chosen like name, public key size, signature size, encryption level, etc.
- generate_keypair()—A function that generates the public and private keys used for the signature scheme.
- export_secret_key()—As the previous function generates both public and private keys as outputs, this function extracts only the private key.
- sign()—A function to perform the signature on the message provided by the user. This is used only on the signer's side.
- verify()—A function to verify if the signature is authentic or not. This is used only by the verifier.

- Class KeyEncapsulation—To enable the KEM algorithms and their functions
  - Kemalg variable—A variable containing the algorithm chosen among "DEFAULT" (A default algorithm in case of no choice), "Classic-McEliece-348864" (Classic McEliece algorithm), "Kyber512" (Crystals Kyber algorithm), "NTRU-HPS-2048–509" (NTRU algorithm), and "LightSaber-KEM" (Saber algorithm).

- details()—A function that displays the details of the algorithm chosen like name, public key size, secret key size, encryption level, etc.
- generate_keypair()—A function that generates the public and private keys used for the signature scheme. This is done on the client's side.
- export_secret_key()—As the previous function generates both public and private keys as outputs, this function extracts only the private key.
- encap_secret()—A function on the server's side that uses the public key as an input to generate a shared secret key and a ciphertext. The shared secret key will henceforth be called shared_secret_server.
- decap_secret()—A function on the client's side that take the ciphertext as the input and produces a shared secret key as the output, henceforth called shared_secret_client.

In addition to the KeyEncapsulation class, the AES encryption library also has to be imported and its functions must be used for encryption.

There are two Python files established for the API. One operates as the client for KEM and as the signer for digital signature algorithms (called client.py) and the other operates as the server for KEM and as the verifier for digital signature algorithms (called server.py). The functionalities and processes are established as mentioned in the methodology section between the two files. It must be noted that each file has two separate sections to deal with KEM and digital signature. The entire system is abstracted well to not allow the interference of any other algorithm in the midst of one being executed at that instance.

The methodology is referred to and the corresponding variables/functions provided in the OQS package are utilized. Since the functions and the codified version are defined by the LibOQS library, the only action that must be taken is to ensure the right variables and information is being fed to the functions.

Web Implementation

The web application is deployed using Python Flask. Flask is a web application framework designed to deploy applications easily using Python and make them scalable. Just installing the package helps use its utilities on many protocols, especially TLS [33, 34].

The API is connected to two HTML files, named sig.html and kem.html for digital signature and KEM respectively. A third HTML file called index.html is the landing page of the application and allows the user to choose between the two categories of algorithms, KEM or Digital Signature. The page gets directed to the corresponding web page as per the choice.

- Sig.html
  - Inputs for the web page are:

Message to be signed
Algorithm selection dropdown menu

– Outputs of the web page are:

The original message
The message length
The digital signature algorithm chosen
The validity of the signature (whether the signature is valid or not)
The signature appended to the message in encrypted form

- kem.html

– Inputs for the web page are:

Message to be communicated
Algorithm selection dropdown menu

- Outputs of the web page are:

The original message
The message length
The KEM algorithm chosen
The Validity of the transfer (Whether there is an intrusion or not)
The secret key used for the encryption
The encrypted message
The decrypted message

The entire framework is finally Dockerized to allow it to be deployed on all platforms that support Docker. To run Docker, the Docker package must be installed. It must be built and a container image must be initiated. The container can then be run and launch the application.

In this work, a WSGI server is established to launch the Flask service on HTTPS and TLS [35]. A Web Server Gateway Interface (WSGI) server has a mechanism to configure server communication with its clients and its response to the clients' requests. The configuration has been made to allow multiple client requests at once. Depending on the CPU power and RAM capacity of the system running the server, multiple processes can be stemmed where each process can serve a client request. The systems used here for development could spawn up to 8 processes, and hence, 8 client requests could be handled in a single instance. The entire system is then made available for testing using NGROK which allows the server to be tunneled to HTTP/HTTPS requests made on a local system.

The algorithms use TLS 1.3 by default. Hence, that version is utilized by web browsers to launch the web application.

Results

The result of the work is that communication is established between server and client using OpenSSL and TLS, where data is encrypted using quantum-safe algorithms and sent across nodes using HTTPS.

On launching, the web application shows the index.html page as in Fig. 8. It offers the user the option of going ahead with either the KEM scheme or the signature scheme.

Figure 9 shows that on choosing the signature scheme, the sig.html page is displayed. The first two fields are input fields that take in the message and the algorithm choice from the user. The API is called and it communicates between the signer and the verifier. The remaining fields on the page are the outputs for the scheme. It must be noted that the signature sizes, encryption scheme, and other features change for each algorithm.

Figure 10 shows that on choosing the KEM scheme, the kem.html page is displayed. The first two fields are input fields that take in the message and the algorithm choice from the user. The API is called and it performs end-to-end encryption using both AES cryptography and quantum-safe KEM algorithms between the server and the client. The remaining fields on the page are the outputs for the scheme. It must be noted that the key sizes, encryption scheme, and other features change for each algorithm.

The web browser's security tabs are opened and searched to test if the web application uses TLS for communication. As evident from Fig. 11, the application runs on TLS 1.3 and also uses AES encryption on the HTTPS connection. The page says that it has valid and secure HTTPS because NGROK generates authentic certificates and keys to establish the connection.

The WSGI server also records the processes that are running the different client requests and the web page of the framework is being accessed. The server records are shown in Fig. 12.

Hence, it can be verified that the application is indeed running on TLS and that the research objective has been satisfied. Furthermore, the encryption shows that the quantum-safe algorithms work efficiently and produce validity as true. Thus,

Fig. 8 Web application launch page

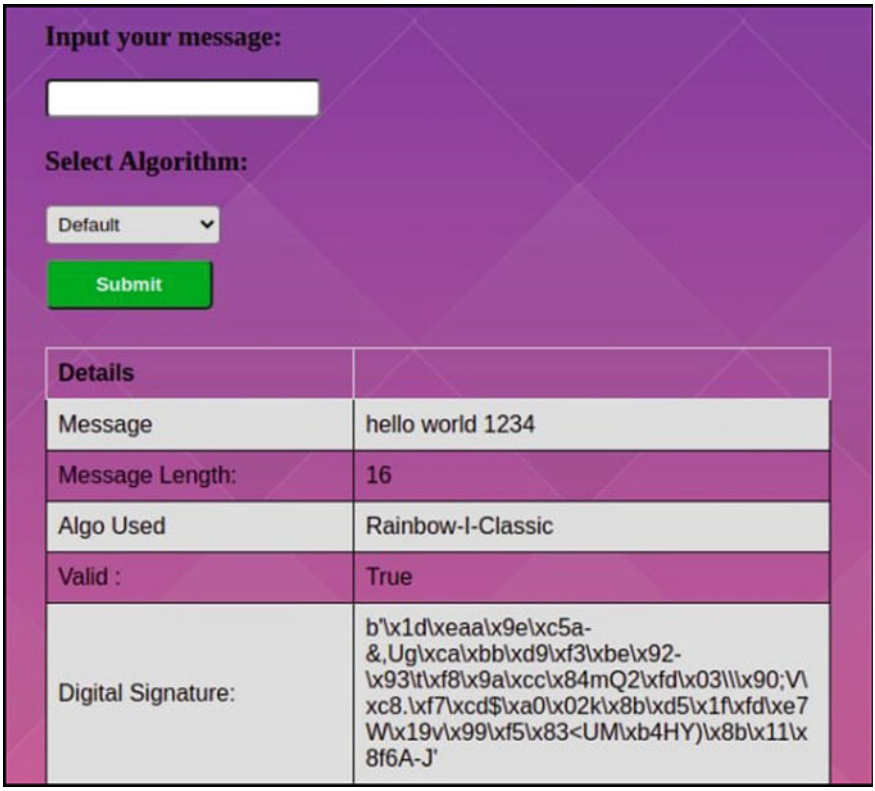
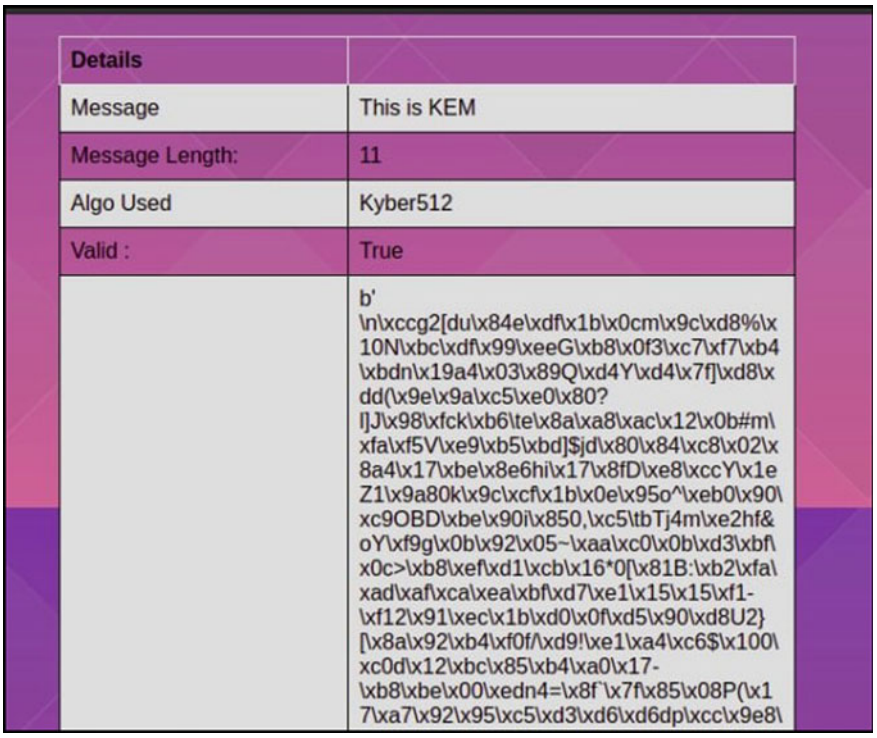


Fig. 9 Web page for digital signature scheme

an efficient framework has been established that satisfies the research objectives completely [36].

Thus, it can be inferred with this research that quantum-safe algorithms can be run effectively using TLS 1.3 protocol and can have authentic communication through web-based applications. The NIST shortlisted algorithms' functionality and working have been defined by LibOQS and this work has utilized their library. The findings are that all shortlisted algorithms can be run efficiently and using the NGROK server, multiple algorithms can be run simultaneously as well, without getting tangled with one another. Also, depending on the size of the message and the chosen algorithm, the key generated changes as well, proving that the input is being taken into account correctly. Hence, if this work is to be used for the information supplied through forms and other mediums, the data would be protected at all times.



Details	
Message	This is KEM
Message Length:	11
Algo Used	Kyber512
Valid :	True
	b' \n\xccg2[du\x84e\xdf\x1b\x0cm\x9c\xd8%\x 10N\xbc\xdf\x99\xeeG\xb8\x0f3\xc7\xf7\xb4 \xbdn\x19a4\x03\x89Q\xd4Y\xd4\x7f]\xd8\x dd(\x9e\x9a\xc5\xe0\x80? l]J\x98\xfck\xb6\te\x8a\xa8\xac\x12\x0b#m\ xfa\xf5V\xe9\xb5\xbd]$jd\x80\x84\xc8\x02\x 8a4\x17\xbe\x8e6hi\x17\x8fD\xe8\xccY\x1e Z1\x9a80k\x9c\xcf\x1b\x0e\x95o^\xeb0\x90\ xc9OBD\xbe\x90i\x850,\xc5\tbTj4m\xe2hf& oY\xf9g\x0b\x92\x05~\xaa\xc0\x0b\xd3\xbf\ x0c>\xb8\xef\xd1\xcb\x16*0[\x81B:\xb2\xfa\ xad\xaf\xca\xea\xbf\xd7\xe1\x15\x15\xf1- \xf12\x91\xec\x1b\xd0\x0f\xd5\x90\xd8U2} [\x8a\x92\xb4\xf0f/\xd9!\xe1\xa4\xc6$\x100\ xc0d\x12\xbc\x85\xb4\xa0\x17- \xb8\xbe\x00\xedn4=\x8f`\x7f\x85\x08P(\x1 7\xa7\x92\x95\xc5\xd3\xd6\xd6dp\xcc\x9e8\

Fig. 10 Web page for KEM scheme

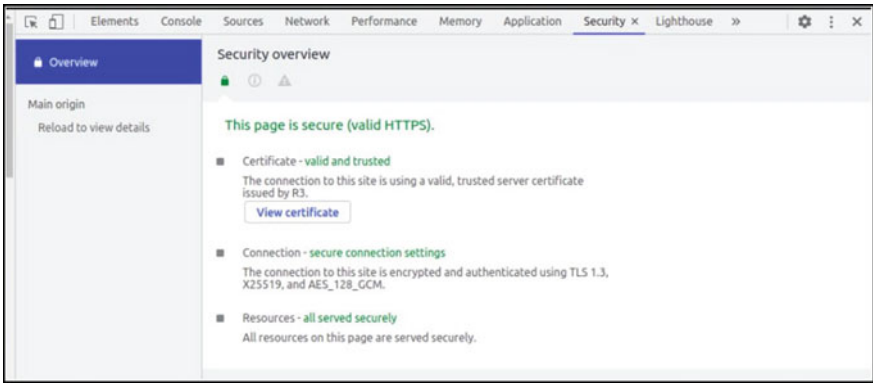


Fig. 11 Security tab of the web browser

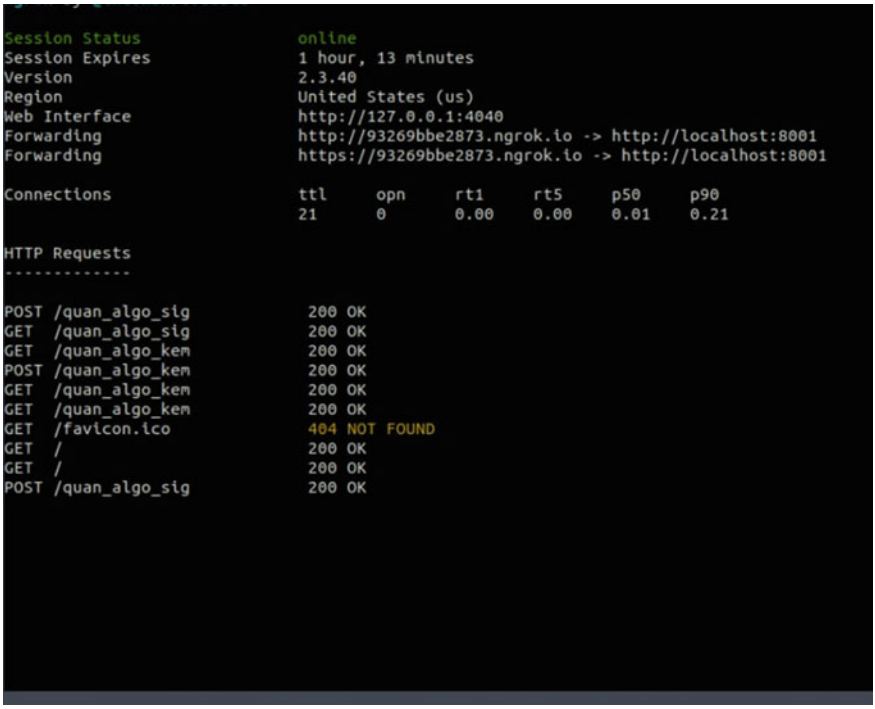


Fig. 12 Server record showing details of multiple requests

Conclusion and Future Work

The research paper focuses on developing a framework that utilizes quantum-safe algorithms on the TLS protocol. The framework takes the help of the LibOQS-Python library to get functions of the algorithms and uses them on a web application to provide end-to-end communication either for digital signature or KEM. To establish TLS handshaking, Flask framework is used with a TLS certificate and key and an HTTPS connection on the web application is established. The user can input the message to be communicated and choose an algorithm depending on the scheme. The output is provided accordingly. The entire framework is Dockerized to allow cross-platform execution.

This implies that a framework has been established using TLS protocol such that all data being communicated via a web application is being encrypted at the client and is decrypted at the server. In the case of digital signature, the data received by the verifier is authentically signed by the signer. Furthermore, unlike the technologies available currently, this framework is capable of executing all the 7 NIST-shortlisted algorithms, perhaps even simultaneously, if multiple clients are deployed.

As a future prospect for this work, it can be extended to include a wider range of applications like IoT, blockchain, or cloud computing. After NIST concludes its

competition of quantum-safe cryptography, the selected standard can be established as the only algorithm of the framework to make it an authorized system for industrial usage. More additions can be made to include encryption systems other than AES, like RSA, in the KEM scheme of encryption. The progress in the field of quantum computing would further bring enhanced capabilities in the future to test the framework in a real-world situation.

Acknowledgements The research team sincerely thanks the team of Unisys India Pvt. Ltd, especially Mr. Anees Ahmed, for giving the opportunity to work on "Application of Quantum Algorithms for Network Protocols" and for providing guidance and support during its development. Special thanks to the mentors of the research, Dr. Thippeswamy M.N., and Mr. Vinay T.R. for their valuable insights. The team also extends gratitude to Nitte Meenakshi Institute of Technology for supporting with the required resources and for helping the idea materialize and succeed.

References

1. Frankenfield J (2019) An article on "Quantum Computing. [Online] Available: Investopedia, https://www.investopedia.com/terms/q/quantum-computing.asp. Accessed 3 Dec 2020
2. An article on "Quantum Superposition". [Online] Available: Joint Quantum Institute, https://jqi.umd.edu/glossary/quantum-superposition. Accessed 10 June 2021
3. Pieper J, Lladser ME (2018) Quantum computation. Scholarpedia. [Online] Available: Scholarpedia, http://www.scholarpedia.org/article/Quantum_Computation#Quantum_Interference_and_Decoherence. Accessed 10 June 2021
4. Orzel C (2017) An article on "How do you create quantum entanglement?. [Online] Available: Forbes, https://www.forbes.com/sites/chadorzel/2017/02/28/how-do-you-create-quantum-entanglement/?sh=6d0ec4ea1732. Accessed 10 June 2021
5. An article on "Quantum Computing 101". [Online] Available: University of Waterloo, https://uwaterloo.ca/institute-for-quantum-computing/quantum-computing-101. Accessed 3 Dec 2020
6. Sihare SR et al (2017) Analysis of quantum algorithms with classical systems counterpart. MECS, IJIEEB 9:20. https://doi.org/10.5815/ijieeb.2017.02.03
7. An initiative on "Post-Quantum Cryptography" (2020) [Online] Available: NIST Computer Security Resource Centre, https://csrc.nist.gov/projects/post-quantum-cryptography. Accessed 3 Dec 2020
8. Aaronson S (2013) Quantum computing since democritus. Cambridge University Press, ISBN 978-0-521-19956-8. Accessed 25 Nov 2020
9. Rieffel E, Polak W (2011) Quantum computing: a gentle introduction. MIT Press. Accessed 25 Nov 2020
10. Sutor RS (2019) Dancing with qubits. Packt Publications. ISBN 978-1-838-82736-6. Accessed 21 Jan 2020
11. Grau A (2020) An article on "Quantum-safe cryptography". [Online] Available: Sectigo, https://sectigo.com/resource-library/quantum-safe-cryptography-surviving-the-upcoming-quantum-cryptographic-apocalypse. Accessed 25 Nov 2020
12. Hoursanov A (2020) an article on "Quantum-safe security". [Online] Available: SAP Innovation Center Network https://icn.sap.com/article/quantum-safe-security-future-proof-data-protection. Accessed 25 Nov 2020
13. Campagna M et al (2015) Quantum-safe cryptography and security. ETSI White Paper No 8. [Online] Available: ETSI https://www.etsi.org/images/files/ETSIWhitePapers/QuantumSafeWhitepaper.pdf. Accessed 25 Nov 2020

14. Lisa et al (2018) Cryptography in a post-quantum world. In: Quantum cryptography whitepaper, Version 5. [Online] Available: Accenture https://www.accenture.com/_acnmedia/PDF-87/Accenture-809668-Quantum-Cryptography-Whitepaper-v05.pdf. Accessed 25 Nov 2020
15. Lyubashevsky V et al (2020) "Security & Privacy", an article in quantum-safe cryptography. [Online] Available: IBM Zurich https://www.zurich.ibm.com/securityprivacy/quantumsafecryptography.html. Accessed 25 Nov 2020
16. "Quantum-Safe Security Position Paper" by Quantum-Safe Security Working Group, 2016. [Online] Available: Cloud Security Alliance https://downloads.cloudsecurityalliance.org/initiatives/qss/What_is_Quantum_Safe_Security_position_paper.pdf. Accessed 25 Nov 2020
17. "Quantum Security Technologies" in White Paper Version 1.0, 2020. [Online] Available: National Cyber Security Center UK https://www.ncsc.gov.uk/whitepaper/quantum-security-technologies. Accessed 25 Nov 2020
18. Easterbrook K et al (2020) Post-quantum TLS. [Online] Available: Microsoft https://www.microsoft.com/en-us/research/project/post-quantum-tls/. Accessed 25 Nov 2020
19. Sikeridis D et al (2020) Post-quantum authentication in TLS 1.3: a performance study. In: NDSS 2020. [Online] Available: Cryptology ePrint Archive https://eprint.iacr.org/2020/071. Accessed 4 Sept 2021
20. Sikeridis D et al (2020) Assessing the overhead of post-quantum cryptography in TLS 1.3 and SSH. In: CoNEXT 2020. [Online] Available: ACM https://doi.org/10.1145/3386367.3431305. Accessed 4 Sept 2021
21. Crockett E et al (2019) Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH. In: NIST conference. [Online] Available: https://www.douglas.stebila.ca/research/papers/NISTPQC-CroPaqSte19/. Accessed 4 Sept 2021
22. Alagic G et al (2020) Status report on the second round of the NIST post-quantum cryptography standardisation process. NISTIR 8309. [Online] Available: NIST Computer Security Resource Center https://csrc.nist.gov/publications/detail/nistir/8309/final. Accessed 25 Nov 2020
23. Stebila D et al (2021) Open quantum-safe project. [Online] Available: Open Quantum-Safe, https://openquantumsafe.org/. Accessed 20 May 2021
24. An article on "Quantum Key Distribution" (2021) [Online] Available: IDQ Antique, https://www.idquantique.com/quantum-safe-security/overview/quantum-key-distribution/. Accessed 10 June 2021
25. Rescorla E (2018) Standards track—the transport layer security (TLS) protocol version 1.3. RFC 8846, ISSN 2070-1721. [Online] Available: IETF, https://datatracker.ietf.org/doc/html/rfc8446. Accessed 10 June 2021
26. OpenSSL Home (2018) [Online] Available: OpenSSL, https://www.openssl.org/. Accessed 15 May 2021
27. Stebila D et al. (2020) LibOQS Python. [Online] Available: GitHub https://github.com/open-quantum-safe/liboqs-python. Accessed 26 Mar 2021
28. Yegulalp S et al (2021) What is Docker? The spark for the container revolution. [Online] Available: InfoWorld https://www.infoworld.com/article/3204171/what-is-docker-the-spark-for-the-container-revolution.html. Accessed 26 Mar 2021
29. Stebila D et al (2021) OQS-OpenSSL_1_1_1. [Online] Available: https://github.com/open-quantum-safe/openssl. Accessed Mar 26, 2021
30. XML Security Working Group F2F (2009) Key encapsulation: a new scheme for public-key encryption. [online] Available: W3 Lists https://lists.w3.org/Archives/Public/public-xmlsec/2009May/att-0032/Key_Encapsulation.pdf. Accessed Mar 2021
31. Lutkevich B et al (2021) What is a digital signature?. [Online] Available: Tech Target https://searchsecurity.techtarget.com/definition/digital-signature. Accessed 10 Jun 2021
32. Stebila D et al (2021) LibOQS. [Online] Available: https://github.com/open-quantum-safe/liboqs. Accessed 26 Mar 2021
33. Pallets (2021) Flask documentation. [Online] Available: https://flask.palletsprojects.com/en/1.1.x/quickstart/. Accessed May 2021
34. Grinberg M (2021) Running your flask application over HTTPS. [Online] Available: https://blog.miguelgrinberg.com/post/running-your-flask-application-over. Accessed Apr 2021

35. Narendra Harny (2021) Implementing a web-socket using flask. [Online] Available: https://medium.com/swlh/implement-a-websocket-using-flask-and-socket-io-python-76afa5bbeae1. Accessed Apr 2021
36. Beullens W et al (2021) Post-quantum cryptography—current state and quantum mitigation. ENISA. [Online] Available: https://www.enisa.europa.eu/publications/post-quantum-cryptography-current-state-and-quantum-mitigation. Accessed May 2021