CRYSTALS-Kyber

Stephen Tambussi
stambussi@scu.edu, 00001469512
COEN251 Network Security, Winter 2023
Santa Clara University

***Abstract*—In this paper, the author defines the post-quantum cryptographic algorithm, CRYSTALS-Kyber. The need for quantum-secure algorithms is discussed along with the history and development of Kyber. The core technologies and algorithmic procedure of Kyber are explained in detail. A comparison with 2 popular pre-quantum public-key algorithms is provided and the potential use cases of Kyber are explored. We look at the shortcomings of the Kyber algorithm and its usage by current companies. Finally, future trends of the Kyber algorithm are reviewed.**

***Keywords*—Key Encapsulation, Key Exchange, Kyber, Post-Quantum Cryptography, Public-key Encryption, Security, Symmetric Encryption**

I. Introduction: What is Kyber?

Kyber is a key encapsulation mechanism (KEM) designed to be resistant to cryptanalysis by quantum computers [1] [2]. Kyber is one of two cryptographic primitives for the Cryptographic Suite for Algebraic Lattices (CRYSTALS) with the other being Dilithium, a post-quantum digital signature algorithm [3]. Kyber and Dilithium are finalists in the National Institute of Standards and Technology's (NIST) post-quantum cryptography project [4]. The security of Kyber is based on the difficulty of solving the learning-with-errors (LWE) problem over module lattices. As a KEM, Kyber is designed to protect a secret key in transit between two parties so that they can

then use the secret key to securely send messages. This paper will provide an overview of the Kyber algorithm, including its technological foundations and comparison with similar systems, while also covering the importance of quantum-resistant cryptography.

Kyber Logo, Image Credit: CRYSTALS Team

II. The need for Quantum-Resistant Cryptography

NIST created the post-quantum cryptography (PQC) project to standardize the next generation of public-key encryption algorithms that are secure against large-scale quantum computers [4]. The goal of post-quantum cryptography is to develop public-key algorithms that are secure against both quantum and classical computers, while retaining interoperability with existing communication protocols and networks [4]. Quantum computers pose a threat to current public-key systems because of their capability of running Shor's algorithm, an algorithm that finds the prime factors of a given integer [5]. The difficulty of finding prime factors and discrete logarithms are the basis for the security of many popular public-key algorithms, such as RSA, DSA, and ECC, each of which can be quickly broken by a powerful quantum computer running Shor's algorithm. Luckily, a quantum computer with enough computing power to run Shor's algorithm has yet to be built. Nevertheless, it is only a matter of time before the technology needed to create practical quantum computers is developed. Therefore, to prevent the breakdown

of internet security, there is a need to establish new quantum-resistant algorithms for cryptography and Kyber is one such algorithm.

III. History and Development of Kyber

A. NIST PQC Round 1

The history of Kyber begins with NIST's Post-Quantum Cryptography Standardization Call for Proposals in 2017 [6]. The call for proposals initiated round one of the standardization program. Submitted for the first round in the KEM category, Kyber competed against a variety of other lattice-based cryptographic algorithms. Each subsequent round of the program would have fewer candidates than the last as NIST selected the best from each round, while also providing comments on each submission. The first round submission for Kyber is considered a rough draft of the final algorithm and as such, changes were made in subsequent rounds to adhere to suggestions made by NIST.

B. NIST PQC Round 2

Beginning in 2019, round two of the standardization program was announced with Kyber selected as one of the candidates [6]. For the second round submission, the Kyber algorithm was modified to adhere to comments made by NIST. One of the primary concerns raised by NIST was that the security proof for Kyber did not directly apply to the algorithm itself, but rather to an altered version of the scheme without the public-key compression [7]. Therefore, the round two Kyber algorithm no longer compressed the public-key, but this introduced a side-effect of increased bandwidth requirements. To balance out the increased bandwidth from removing public-key compression, an algorithmic parameter was reduced without impacting security.

Other changes made to the algorithm include parameter and computational tweaks to improve performance [8].

C. NIST PQC Round 3 and Beyond

NIST announced the seven finalists for the third and final round of the standardization program in the middle of 2020. The algorithms included in this round hold the most promise and are considered for standardization at the round's conclusion [6]. The Kyber algorithm incorporated minor changes during this round, consisting of more parameter tweaks and efficiency improvements [9]. Ultimately, in mid-2022, NIST announced Kyber as one of the winners for the standardization program, certifying its role as a post-quantum standard.

IV. CORE TECHNOLOGIES OF KYBER

Kyber incorporates various technologies that differentiate it from existing cryptographic algorithms. As a key encapsulation mechanism (KEM), Kyber provides confidentiality by securing symmetric key material in a different way from classical public-key encryption schemes. Kyber also offers the highest level of ciphertext indistinguishability, which is a property of many cryptosystems. The security of Kyber is based on the learning-with-errors (LWE) problem over module lattices to provide protection against attacks by both classical and quantum computers.

A. Key Encapsulation Mechanisms

Key encapsulation mechanisms are a type of key exchange that are used to secure the transmission of symmetric key material through public-key algorithms [10]. Similar to public-key encryption (PKE) schemes, the primary application of KEMs is for establishing a shared symmetric key between two parties, which provides confidentiality. However, there is a

distinct difference between the two schemes. Rather than creating a symmetric key and padding it for transmission, the KEM process instead generates a random element from the public key and then uses this element to derive the symmetric key with a key derivation function (KDF). A KDF is a one-way function that takes as input the random element and outputs the symmetric key [11]. Furthermore, unlike PKE schemes, the secret value of the KEM is the randomly generated element and not the padded symmetric key. The process is as follows, the secret value is first generated and used to derive the symmetric key at the sender. Then, it is encrypted with the recipient's public key, transmitted to the recipient, and decrypted with the recipient's private key. Finally, the recipient derives the symmetric key by using the KDF on the secret value [10]. This KEM process (Figure 1) can be summarized as a collection of three algorithms: Generate, Encapsulate, and Decapsulate [12].

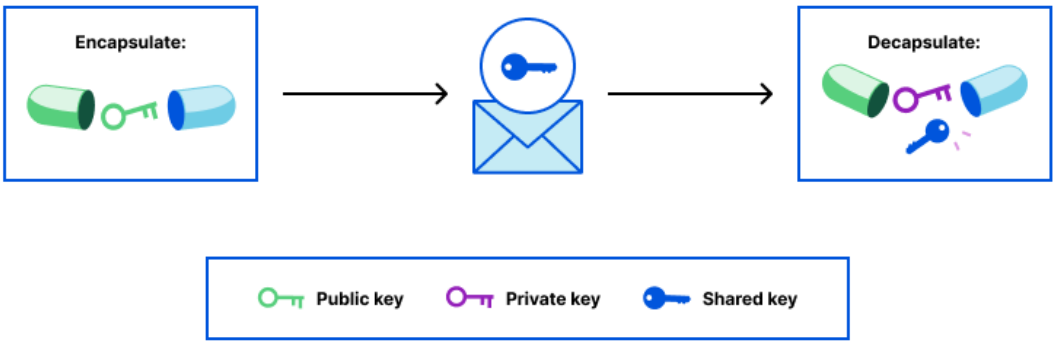

Figure 1. Key Encapsulation Mechanism

B. Ciphertext Indistinguishability

Ciphertext indistinguishability is a property of cryptosystems that prevents adversaries from determining pairs of ciphertexts based on the message they encrypt [13]. There are three definitions of this indistinguishability, with each one being more secure than the last: indistinguishability under chosen plaintext attack (IND-CPA), indistinguishability under chosen

ciphertext attack (IND-CCA1), and indistinguishability under adaptive chosen ciphertext attack (IND-CCA2). In IND-CPA, an adversary generates two messages of equivalent length, a challenger randomly encrypts one of them to make the challenge ciphertext, and then the adversary tries to guess which message was encrypted based on the challenge ciphertext. For IND-CCA1, the general process is similar to IND-CPA, except that the adversary can now encrypt or decrypt random messages before guessing which message was encrypted. IND-CCA2 adds an additional element to IND-CCA1: the adversary now has access to the challenge ciphertext, which cannot be decrypted, and the ability to encrypt or decrypt random messages before guessing. Kyber is classified as a IND-CCA2 secure algorithm and therefore has the strongest possible security [1].

C. Learning-with-errors over Module Lattices

The learning-with-errors problem is a generalization of the parity learning problem first introduced by Oded Regev [1] [14] [15]. The concept behind LWE is the representation of secret information as a collection of equations with errors or, in other words, hiding a secret value by introducing noise to it [16]. The LWE problem is theorized to be difficult to solve using both classical and quantum computers and therefore proves useful in post-quantum cryptography. Kyber's security is based on the hardness of solving a variation of the LWE problem, known as the LWE problem over module lattices (Module-LWE) [15].

V. Overview of the Kyber Algorithm

The Kyber algorithm uses a two-stage approach to achieve its security capabilities. The first stage is a IND-CPA-secure public-key encryption scheme called Kyber.CPAPKE. The second stage applies a variant of the Fujisaki-Okamoto transform to the first stage to create a

IND-CCA2-secure KEM, named Kyber.CCAKEM [9]. The result of these two stages can then be used to construct IND-CCA2-secure public-key encryption (PKE), key exchange, and authenticated-key-exchange schemes (Figure 2) [15] [17]. Similar to public-key schemes, Kyber includes a public key for encryption and a private key for decryption. There are three different parameter sets for Kyber that each provide increasing levels of security: Kyber-512, Kyber-768, and Kyber-1024.

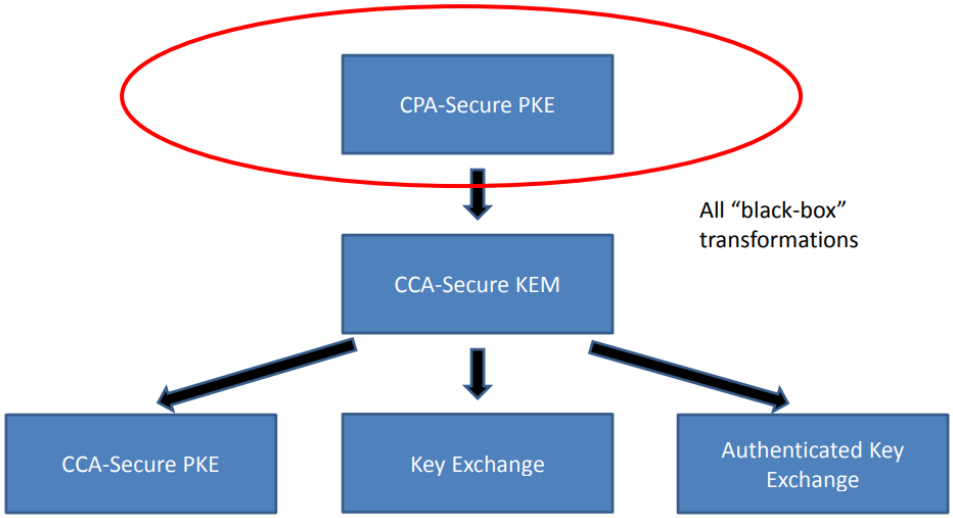

Figure 2. Stages of Kyber Algorithm

A. Key Information

Following the second-stage of Kyber's construction, the public and private keys are fully defined and can be used for the exchange of symmetric key material to establish secure communications. Both the public and private key consist of multiple elements. The public key consists of two elements: **t**, an encoded vector of polynomials and **rho**, the public seed [18]. The private key consists of three elements: **s**, an encoded sample from a centered binomial distribution, **H(pk)**, the hashed public key, and **z**, a nonce [18].

B. Kyber.CPAPKE

The first stage of the Kyber algorithm is a public-key encryption scheme named Kyber.CPAPKE. This stage defines three functions: key generation, encryption, and decryption. Each of these functions are then used in the second stage to construct the KEM.

1) Key Generation

The key generation function creates the public and private key for this stage. This function takes no inputs and outputs the public key **pk**, and the private key **sk**. The generation of the public key **pk** is a multistep process. First, a random matrix **A** is derived from an extendable output function (XOF) using the value of **rho**, the public seed. Then, a vector of polynomials **s**, is sampled from a centered binomial distribution on the output of a pseudorandom function (PRF) [9]. Next, another vector of polynomials **e**, known as the error vector, is sampled in the same way that **s** was sampled [19]. These values are then used to compute the vector of polynomials **t** via the following equation: $\mathbf{t} = \mathbf{As} + \mathbf{e}$. Finally, the result of the modulus operation on vector **t** with parameter q is encoded and assigned to the value of the public key **pk** along with the public seed **rho**.

Following the creation of the public key, the private key **sk** is simply the encoded result of the modulus operation on vector **s** with parameter q.

2) Encryption

Encryption is performed by using the public key. The function takes as input, the public key **pk**, a plaintext message, and a random value. It outputs the ciphertext of the plaintext message. The encryption procedure is similar to the steps for key generation, but with a few differences. First, the plaintext message is converted into a polynomial **m** by using the bits of its binary representation as coefficients [19]. The message polynomial is then scaled by a factor of parameter q. Next, the vector **t** is decoded from the public key **pk**. Then, a matrix $\mathbf{A}^{\mathbf{T}}$ is derived

from an extendable output function (XOF) using **pk** as input. Following this, two polynomial vectors, **r** and $\mathbf{e_1}$, are sampled from a centered binomial distribution on the output of a pseudorandom function (PRF). The PRF takes as input the random value. A polynomial $\mathbf{e_2}$ is also sampled from a centered binomial distribution. The results of the previous steps are now used to compute two ciphertext values, **(u, v)** [9]. The polynomial vector **u** is computed via the following equation: $\mathbf{u} = \mathbf{A^T r} + \mathbf{e_1}$. Then, the polynomial **v** is calculated using: $\mathbf{v} = \mathbf{t^T r} + \mathbf{e_2} + \mathbf{m}$. The final step is the compression of these two ciphertext values [9].

3) Decryption

Decryption is performed by using the private key. The input for the decryption function is the private key **sk** and the ciphertext from the encryption function. It outputs the original plaintext message that was input to the previous encryption function. First, the two ciphertext values **(u, v)** are decompressed [9]. Then, the vector **s** is decoded from the private key **sk**. Next, the two ciphertext values and the private key are used to compute a noisy result of the original plaintext: $\mathbf{m_n} = \mathbf{v} - \mathbf{s^T u}$. This noisy result is decoded to reveal the scaled message polynomial. Finally, the message polynomial is scaled down by the reciprocal of the factor of parameter *q* to recover the plaintext message [19].

C. Kyber.CCAKEM

In the second stage of the Kyber algorithm, known as Kyber.CCAKEM, the key encapsulation mechanism is formed. This stage defines three functions: key generation, encapsulation, and decapsulation. Each of these functions use the functions previously defined in the Kyber.CPAPKE stage.

1) Key Generation

Key generation in this stage creates the public and private key for the KEM. This function takes no inputs and outputs the public key **pk**, and the private key **sk**. The public key is simply the value returned from calling the key generation function of Kyber.CPAPKE. The private key now consists of multiple elements: **s**, the returned value from Kyber.CPAPKE's key generation function, **H(pk)**, a hash of the public key, and **z**, a nonce value [9].

2) Encapsulation

Encapsulation is performed by using the public key. This function takes as input the public key **pk**. The output is a ciphertext of the symmetric key material **c** and the symmetric key **K**. First, the symmetric key material **m** is generated by creating a random value and hashing it. The value **m** and a hash of the public key, **H(pk)**, are then used as input to another hash function. The output of this hash function is the pre-key value **K'** and a random value **r**. Next, the encryption function of Kyber.CPAPKE is used to generate a ciphertext of the symmetric key material, **c**. It takes as input the public key, **pk**, the symmetric key material **m**, and the random value **r**. Finally, the symmetric key **K** is derived by using a KDF on the pre-key value **K'** and a hash of the previously created ciphertext, **H(c)** [9].

3) Decapsulation

Decapsulation is performed by using the private key. This function takes as input the ciphertext of the symmetric key material **c** and the private key, **sk**. The output is the symmetric key **K**. First, the public key **pk**, hash of the public key **h**, and the nonce value **z** are all derived from the private key **sk**. Then, the symmetric key material **m** is recovered by using the decryption function of Kyber.CPAPKE on the ciphertext **c**. Next, the decrypted symmetric key material **m** and the hash of the public key **h** are taken as input to a hash function to generate the pre-key value **K'** and random value **r**. Following this, the encryption function of Kyber.CPAPKE

is used to generate an additional ciphertext of the symmetric key material, **c'**. It takes as input the public key, **pk**, the symmetric key material **m**, and the random value **r**. The two ciphertexts, **c** and **c'**, are compared and if they are equivalent, then the symmetric key **K** is derived from the result of the KDF on the pre-key value **K'** and a hash of the symmetric key material ciphertext, **H(c)**, the same procedure as encapsulation. However, if they are not equivalent, then the symmetric key **K** is derived from the result of the KDF on the nonce value **z** and the hash of the symmetric key material ciphertext, **H(c)** [9]. Therefore, due to the method by which the Kyber KEM is constructed, there is a possibility that the decapsulation operation fails [9]. However, the probability of this occurring is so small that it is negligible.

D. Parameter Sets

Kyber offers three different parameter sets that provide differing levels of security. The security of Kyber-512 is approximately equivalent to AES-128, Kyber-768 is equivalent to AES-192, and Kyber-1024 is the same as AES-256 [1]. Each of these parameter sets modifies the values for the parameters and the public and private key sizes. The following table (Table 1) lists the key sizes, ciphertext size, and parameter values for each parameter set.

TABLE I
KYBER PARAMETER SETS [9]

Version	Size in Bytes			Values						
	Public Key	**Private Key**	**Ciphertext**	n	k	q	η_1	η_2	(d_u, d_v)	δ
Kyber-512	800	1632	768	256	2	3329	3	2	(10, 4)	2^{-139}
Kyber-768	1184	2400	1088	256	3	3329	2	2	(10, 4)	2^{-164}
Kyber-1024	1568	3168	1568	256	4	3329	2	2	(11, 5)	2^{-174}

The parameter sets primarily influence the security of Kyber by changing the size of the values used in the computations of the algorithm. Parameter n determines the maximum degree

of the polynomials used in the calculations. The parameter $\boldsymbol{k}$ sets the number of polynomials per vector, while $\boldsymbol{q}$ is the modulus for numbers. Parameters η_1 and η_2 control the amount of noise in the error polynomials [9]. The values of (d_u, d_v) determine the amount of compression to perform on the output of the Kyber.CPAPKE encryption function, (**u**, **v**). Finally, δ indicates the probability of a decapsulation operation failing [19].

1) 90s Variant

The specification for Kyber includes an alteration of the main design known as the 90s variant. This variant retains the same values for its parameter sets as the default Kyber algorithm and only changes the internal symmetric primitives. Furthermore, it is significantly faster due to its usage of symmetric primitives accelerated in hardware on a large variety of platforms [9]. The default version of Kyber uses SHAKE-128 for the XOF, SHA3-256 and SHA3-512 for its hash functions, and SHAKE-256 for the PRF and KDF. SHAKE is in the SHA-3 family of algorithms. The 90s variant of Kyber uses the CTR mode of AES-256 for the XOF and PRF, SHA-256 and SHA-512 for the hash functions, and SHA-256 for the KDF [9]. These symmetric primitives are implemented in hardware on most recent Intel, AMD, and ARM processors enabling almost twice the performance of the default Kyber algorithm [9].

VI. Comparison of Public-Key Cryptosystems

Most public-key cryptosystems are not used to directly encrypt transmitted data as they are too slow for practical applications. Rather, public-key cryptosystems are used to establish a shared secret key between two parties for symmetric encryption of the data to be transmitted; these are known as hybrid cryptosystems. Similarly, Kyber is a KEM that can be used in a hybrid cryptosystem for the exchange of symmetric keys [15]. Therefore, it is important to compare

Kyber with established public-key cryptosystems, such as RSA and ECC, to better understand the differences between classical and post-quantum cryptography.

A. Rivest-Shamir-Adleman

Rivest-Shamir-Adleman (RSA) is one of the oldest and most well-known public-key cryptosystems. It is commonly used for the transmission of shared keys for symmetric encryption [20]. The security of RSA relies on the difficulty of solving the factoring problem, factoring the product of two large prime numbers [20]. As previously stated, this problem is impractically difficult to solve on classical computers, but a sufficiently powerful quantum computer running Shor's algorithm can solve it. For this reason, Kyber was developed with a different mathematical foundation for its security than RSA to enable protection from attacks by quantum computers. Both Kyber and RSA can be used for the exchange of symmetric keys. However, the manner in which they perform this exchange is different. In Kyber, the symmetric key is derived from a random element generated during the key exchange process and a ciphertext of this random element is the element transmitted using the public-key cryptosystem, not the symmetric key itself. For RSA, the symmetric key already exists before the exchange process and it is required to be padded by a padding scheme such as OAEP [20].

B. Elliptic-curve Cryptography

Elliptic-curve Cryptography (ECC) is a public-key cryptosystem that is based on the algebraic structure of elliptic curves over finite fields. It can be used for the transmission of shared keys for symmetric encryption, among other tasks [21]. ECC's security is based on the difficulty of finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point [21]. While it is infeasible for a classical computer to compute the discrete logarithm, a powerful quantum computer will be able to do so and render ECC insecure.

Although ECC offers greater security at equivalent key sizes when compared to RSA, it still lacks the protection that Kyber provides against quantum attacks. Unlike Kyber, key exchange using ECC requires the symmetric key to already exist before the exchange process, similar to RSA. This particular application of ECC is known as Elliptic-curve Diffie-Hellman (ECDH) and is based on the Diffie-Hellman scheme [21].

VII. USE CASES FOR KYBER

Kyber is a relatively new algorithm and while it has not yet seen widespread use when compared to contemporary algorithms, its adoption will only continue to rise. Kyber has many use cases because it is applicable in any scenario where there is a need to establish a shared symmetric key (Figure 3) or a need for public-key encryption, as it offers both. Furthermore, there are not many barriers to Kyber's continued adoption since it provides a publicly available code repository of its reference implementation written in the C programming language [22]. As a result of this, there have been many third-party implementations of Kyber written in various languages [23]. The following sections will explore some of the potential use cases for the Kyber algorithm.

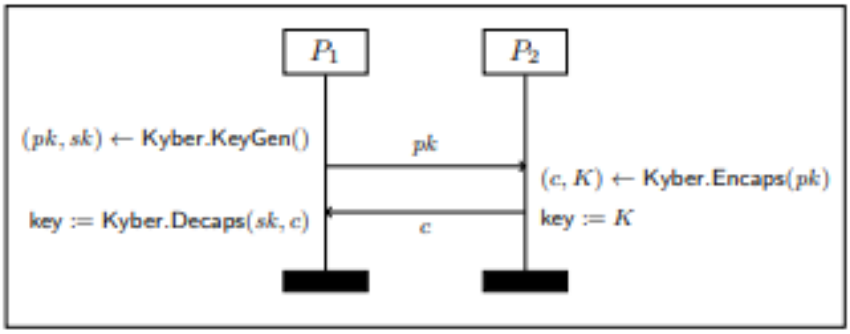

Figure 3. Key Exchange Protocol using the Kyber KEM

A. Securing Network Communications

Transport Layer Security (TLS) is a widely used protocol to secure network communications and is a prime candidate for Kyber. TLS is used in many network applications, including HTTPS. It offers many algorithms in its cipher suite for various operations. One of these operations, the TLS handshake, relies on a key exchange scheme using public-key algorithms. In the TLS handshake, the client encrypts a symmetric key with the public key of the server and then sends it to the server to use as the shared key for the network session [24]. Some of the algorithms used for this handshake are RSA, Diffie-Hellman, and ECDH. Once quantum computers advance to the point of practical usage, Kyber can be a drop-in replacement for these algorithms in TLS as it is a key exchange mechanism. On a related note, previous versions of the TLS handshake algorithms were vulnerable to man-in-the-middle attacks because they did authenticate the server or client [25]. Kyber defines an authenticated version of its key exchange protocol, known as Kyber.AKE, that provides protection against these attacks (Figure 4) [15].

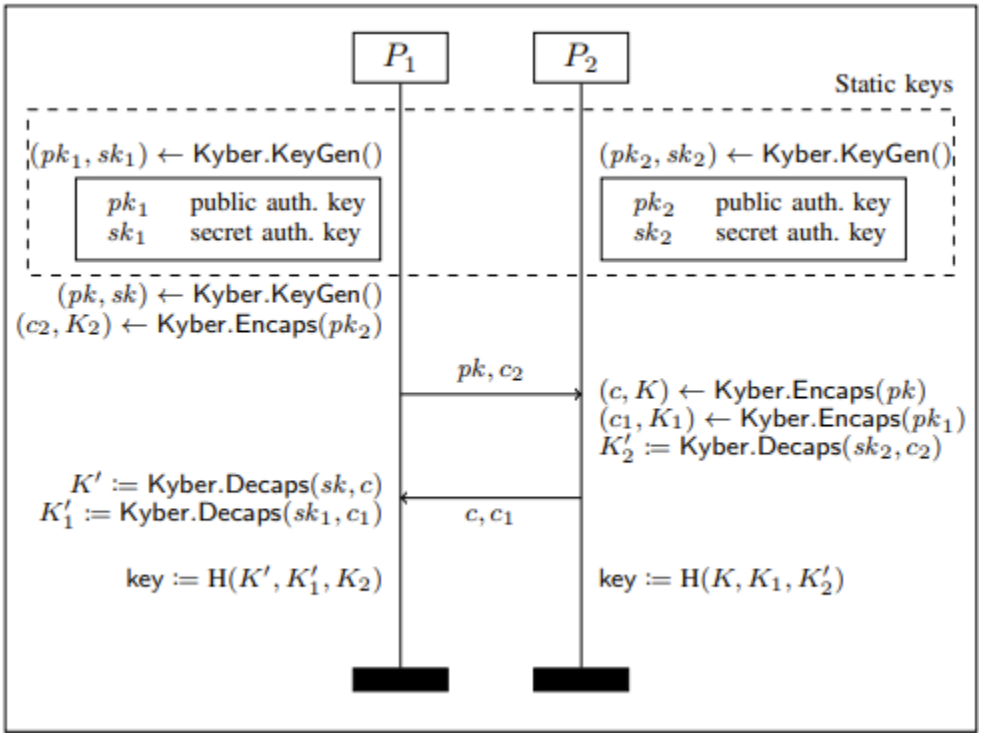

Figure 4. Authenticated Key Exchange Protocol using Kyber

B. Remote Access Security

Similar to securing network communications, remotely accessing computers over a network is another candidate for usage of the Kyber algorithm. There are many protocols for remote access services, but one of the most popular is the Secure Shell Protocol (SSH). SSH primarily enables remote usage of a computer over a network [26]. To secure the remote access session, the SSH protocol encrypts all the communication in the session with a symmetric key. This symmetric key is established between the client and server using a key exchange method. Currently, the algorithms used for this key exchange are either Diffie-Hellman or ECDH [27]. The key exchange procedure of SSH is a prime use case for Kyber. Specifically, the previously defined key exchange protocol with the Kyber KEM could be used in SSH. After the SSH session is securely established, the user is then authenticated. The most commonly used mechanism for user authentication is a password [26]. However, passwords are vulnerable to brute-force attacks so the SSH protocol also offers public-key-based authentication, which uses a set of asymmetric keys to authenticate the user [26]. This form of authentication is another use case for Kyber as it can generate a quantum-resistant key pair in its public-key encryption scheme, Kyber.CPAPKE. Overall, the SSH protocol is an excellent use case for Kyber as it can enable a quantum-secure remote access session for users.

VIII. LIMITATIONS OF KYBER

While the advantages of Kyber are clear, such as its quantum resistance and ease of implementation, its limitations have not yet been discussed. Due to the fact that Kyber is a newer algorithm, there have only been a few potential security vulnerabilities discovered. Typically,

more vulnerabilities are found when cryptographic algorithms begin to be widely used and Kyber is not yet at that stage. Furthermore, Kyber's security is modeled after attacks by hypothetical quantum computers and until such computers exist, it is difficult to fully test the robustness of Kyber's security. Other than potential vulnerabilities, Kyber is limited in its performance. Cryptographic algorithms are computationally costly and Kyber is no exception. Specifically, Kyber is more computationally costly than pre-quantum cryptographic algorithms due to its complex mathematics and use of intensive symmetric primitives.

A. Potential Security Vulnerabilities

1) Side-channel Attack

A vulnerability of Kyber that was discovered and widely publicized is a novel side-channel attack. These types of attacks analyze traces of physical signals, such as timing data and electromagnetic radiation, that computer systems emit to collect secret information [28]. Researchers at the KTH Royal Institute in Sweden developed a method to extract the private key of a particular implementation of Kyber using a newly developed training method for neural networks [28]. However, this side-channel attack was downplayed by NIST as they clarified that it only compromises a particular implementation of Kyber and not the security of the algorithm itself [28]. Nevertheless, this research offers greater insight into the vulnerabilities of Kyber and potential future avenues for cryptanalysis of its algorithm.

2) Attacks against Symmetric Primitives

As previously discussed, Kyber uses symmetric primitives in its two-stage construction of the KEM. One of these symmetric primitives is SHAKE, an extendable-output function (XOF). As an XOF, SHAKE produces an output that appears uniformly random. A potential attack on Kyber would exploit a vulnerability inherent in the SHAKE symmetric primitive. For example, if

a flaw was discovered in the SHAKE algorithm that enables an attacker to find an association between a particular input and output, then this flaw could be exploited to compromise the overarching Kyber algorithm. However, there would need to be a major breakthrough in the cryptanalysis of SHAKE for this attack to be possible [9]. Even so, if the SHAKE algorithm was compromised, then Kyber could be modified by simply replacing SHAKE with another XOF to render this attack useless [9].

B. Performance Considerations

Kyber is computationally expensive when compared to pre-quantum algorithms currently in use. This is partly due to the complexity of its mathematical computations, but more of Kyber's performance is determined by the performance of its symmetric primitives [9]. As part of the evaluation process for the post-quantum cryptography (PQC) project, NIST recommended that the candidate algorithms be tested on a ARM Cortex-M4 reference platform to determine their feasibility on microcontrollers and gain better insight into their overall performance [29]. CPU clock cycle count was the primary metric for performance evaluation. Compared to other competing PQC KEMs, the implementation of Kyber on the ARM Cortex-M4 platform had the lowest cycle count for the sum of the key generation, encapsulation, and decapsulation functions [29]. While Kyber may be the fastest of the PQC KEMs, it is still slower than pre-quantum key exchange algorithms. To better understand the performance limitations of Kyber, we will compare it with a Cortex-M4 implementation of Elliptic-curve Diffie-Hellman (ECDH) using Curve25519, called X25519 [30]. This comparison measures cycle counts (Table 2).

TABLE II
PERFORMANCE COMPARISON OF KYBER AND ECDH (X25519)

	Kyber-512 [9]	**Kyber-768** [9]	**Kyber-1024** [9]	**ECDH(X25519)** [30]
Key Generation	463,068	756,224	1,213,303	*Not measured*
Encapsulation	561,518	915,676	1,407,769	*Not measured*
Decapsulation	519,237	853,001	1,326,409	*Not measured*
Total	**1,543,823**	**2,524,901**	**3,947,481**	**894,391**

For the measurements, both the Kyber and ECDH implementations were optimized for the Cortex-M4. It is also important to note that this Kyber implementation is the default variant with SHAKE and not the 90s variant with different symmetric primitives. Regarding the performance results, Kyber-512, the lowest security level and fastest of the three, is still substantially slower than the ECDH implementation. The performance difference widens further as the security level increases with Kyber-1024 being more than four times as slow as ECDH. This comparison suggests that the usage of Kyber may lead to slower performance in applications unless further optimizations can be made or its default symmetric primitives can be hardware accelerated.

IX. CURRENT USERS OF KYBER

Kyber is a relatively new algorithm and even still there have been a few companies that have implemented the algorithm into their services and products. The following sections will discuss three of the most notable companies using Kyber.

A. Cloudflare

Cloudflare is a technology company that provides performance and cybersecurity services for web applications. They sell security software for websites and related services to make protection easy and accessible. One of their provided services is TLS encryption. Cloudflare has

been doing post-quantum TLS experiments to test new algorithms and introduced their Cloudflare Interoperable Reusable Cryptographic Library (CIRCL). CIRCL is a collection of implementations for cryptographic algorithms written in the Go programming language. Recently, Cloudflare team members have developed high-performance implementations of Kyber and other post-quantum algorithms in CIRCL [31].

B. AWS

Amazon Web Services (AWS) is a subsidiary of Amazon that provides cloud computing services. Amazon announced in a blog post that their Key Management Service (KMS) added three hybrid post-quantum key exchange algorithms for the TLS encryption protocol, which is used when connecting to AWS KMS API endpoints [32]. One of these algorithms is Kyber. The added algorithms are considered hybrid because they combine the security protections of both the classical and post-quantum key exchange algorithms in a single TLS handshake [32]. Furthermore, the addition of Kyber, specifically Kyber-512, to the TLS handshake only adds 0.3 milliseconds of latency overhead when compared to the TLS handshake with the classical algorithm [32]. This decrease in performance is insignificant when compared to the enhanced security afforded by the addition of Kyber.

C. IBM

International Business Machines (IBM) is a technology company that develops computer hardware and software, while also doing research and development of various new technologies. As one of the contributors to the design of Kyber, IBM has already developed a product implementing both Kyber and Dilithium. This product is a quantum-secure tape drive. Magnetic tape drives use a combination of symmetric and asymmetric encryption to secure archival data for many years [33]. To create this quantum-secure drive, IBM used both Kyber and Dilithium

with AES-256 on a state-of-the-art tape drive to enable post-quantum security and ensure that this drive would be secure for the years to come [33]. Existing drives can also benefit from these new algorithms through a firmware update [33].

X. Future Trends of Kyber

Potential future directions for Kyber are more optimized implementations and security improvements.

A. Increasing the Performance of Kyber

Kyber is one of the fastest of the post-quantum cryptographic algorithms. However, as previously discussed, it is still much slower than many pre-quantum algorithms, especially as its security level increases. Furthermore, Kyber's performance is largely determined by its underlying symmetric primitives. Therefore, the focus for future work should be developing optimized Kyber implementations and improving hardware acceleration of its algorithm to reach performance parity with pre-quantum algorithms. One example for this type of future work is that researchers have already developed a novel high-performance implementation of Kyber, which uses an AI accelerator found on newer GPUs to achieve a significant speedup [34].

B. Security Improvements

The highest security level of Kyber is Kyber-1024, which is roughly equivalent to the security provided by AES-256. Currently, this level provides more than sufficient security for most applications. In the future, however, it may become necessary for Kyber to offer even higher levels of security for specific use cases, depending on the advancement of quantum computers. Furthermore, as mentioned before, Kyber can be vulnerable to attacks that exploit flaws in platform-specific implementations, known as side-channel attacks. Future work for

Kyber should focus on designing implementations that are resistant to these types of attacks and any others to further increase security. Researchers have already begun to develop Kyber implementations that are resistant to side-channel attacks, such as a version of Kyber that is implemented entirely in hardware on an FPGA [35].

XI. CLOSING

The Kyber algorithm is the cutting-edge of cryptography, with much research into its design and implications for the security world. With the advent of practical quantum computers, algorithms such as Kyber are increasingly necessary to replace current pre-quantum algorithms. Kyber's algorithm does not rely on needlessly complex procedures to conceal data and rather uses a novel application of module lattices as the basis for its security. The algorithm is also easy to implement and therefore has many applications and use cases. Kyber is still slower than pre-quantum algorithms and has a few points of potential vulnerabilities, with more to be discovered. Even so, this has not stopped companies from beginning to integrate Kyber and its companion algorithm, Dilithium, in new and interesting ways. As time continues, Kyber will only continue to be more widely used as NIST moves to officially standardize it and future work is made to further improve the algorithm. Kyber is the result of many years of research and it will provide security for future applications in the post-quantum era.

XII. REFERENCES

[1] P. Schwabe, "Kyber," *pq-crystals.org*, Dec. 23, 2020. [Online]. Available: https://pq-crystals.org/kyber/index.shtml. [Accessed: Feb. 20, 2023].

[2] Wikipedia Contributors, "Kyber," *wikipedia.org*, Jan. 13, 2022. [Online]. Available: *https://en.wikipedia.org/wiki/Kyber*. [Accessed: Feb. 20, 2023].

[3] P. Schwabe, “CRYSTALS,” *pq-crystals.org*, Feb. 25, 2022. [Online]. Available: https://pq-crystals.org/index.shtml. [Accessed: Feb. 20, 2023].

[4] National Institute of Standards and Technology, “Post-Quantum Cryptography | CSRC,” *National Institute of Standards and Technology*, Post-Quantum Cryptography Project, 2017. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography. [Accessed: Feb. 21, 2023].

[5] Wikipedia Contributors, “Post-quantum cryptography,” *wikipedia.org*, Mar. 18, 2010. [Online]. Available: https://en.wikipedia.org/wiki/Post-quantum_cryptography. [Accessed: Feb. 21, 2023].

[6] Wikipedia Contributors, “NIST Post-Quantum Cryptography Standardization,” *wikipedia.org*, Dec. 29, 2017. [Online]. Available: https://en.wikipedia.org/wiki/NIST_Post-Quantum_Cryptography_Standardization. [Accessed: Feb. 21, 2023].

[7] P. Schwabe. Conference Presentation, Topic: “CRYSTALS-Kyber.” 2nd NIST PQC Standardization Conference, Aug. 23, 2019.

[8] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, “CRYSTALS-Kyber,” Algorithm Specifications And Supporting Documentation (version 2.0), 30 Mar. 2019.

[9] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, “CRYSTALS-Kyber,” Algorithm Specifications And Supporting Documentation (version 3.02), 4 Aug. 2021.

[10] Wikipedia Contributors, “Key encapsulation mechanism,” *wikipedia.org*, Dec. 24, 2006. [Online]. Available: https://en.wikipedia.org/wiki/Key_encapsulation_mechanism. [Accessed: Feb. 25, 2023].

[11] Wikipedia Contributors, “Key derivation function,” *wikipedia.org*, Feb. 1, 2004. [Online]. Available: https://en.wikipedia.org/wiki/Key_derivation_function. [Accessed: Feb. 25, 2023].

[12] G. Tamvada and S. Celi, “Deep dive into a post-quantum key encapsulation algorithm,” *cloudflare.com*, Feb. 22, 2022. [Online]. Available: https://blog.cloudflare.com/post-quantum-key-encapsulation/. [Accessed: Feb. 26, 2023].

[13] Wikipedia Contributors, “Ciphertext indistinguishability,” *wikipedia.org*, Jul. 7, 2005. [Online]. Available: https://en.wikipedia.org/wiki/Ciphertext_indistinguishability. [Accessed: Feb. 26, 2023].

[14] O. Regev, “On lattices, learning with errors, random linear codes, and cryptography,” *Journal of the ACM*, vol. 56, no. 34, p. 1-40, September, 2009. [Online serial]. Available: https://dl.acm.org/doi/10.1145/1568318.1568324. [Accessed Feb. 27, 2023].

[15] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, “CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM”. *In 2018*

IEEE European Symposium on Security and Privacy, EuroS&P 2018. IEEE, 2018. Available: https://eprint.iacr.org/2017/634

[16] V. Lyubashevsky, C. Peikert, and O. Regev, "On Ideal Lattices and Learning with Errors over Rings," *Journal of the ACM*, vol. 60, no. 43, p. 1-35, November, 2013. [Online serial]. Available: https://dl.acm.org/doi/10.1145/2535925. [Accessed Feb. 28, 2023].

[17] V. Lyubashevsky. Conference Presentation, Topic: "Standardizing Lattice Cryptography … and Beyond." PQCRYPTO 2017, June. 28, 2017.

[18] C. Vredendaal, S. Dragone, B. Hess, T. Visegrady, M. Osborne, D. Bong, and J. Bos "Quantum Safe Cryptography Key Information for CRYSTALS-Kyber," *Internet Engineering Task Force*, October, 2022. [Online serial]. Available: https://www.ietf.org/id/draft-uni-qsckeys-kyber-00.html. [Accessed Mar. 1, 2023].

[19] R. Gonzalez, "Kyber - How does it work?," *cryptopedia.dev*, Sep. 14, 2021. [Online]. Available: https://cryptopedia.dev/posts/kyber/. [Accessed Mar. 1, 2023].

[20] Wikipedia Contributors, "RSA (cryptosystem)," *wikipedia.org*, Jul. 26, 2001. [Online]. Available: https://en.wikipedia.org/wiki/RSA_(cryptosystem). [Accessed: Mar. 1, 2023].

[21] Wikipedia Contributors, "Elliptic-curve cryptography," *wikipedia.org*, Oct. 23, 2001. [Online]. Available: https://en.wikipedia.org/wiki/Elliptic-curve_cryptography. [Accessed: Mar. 1, 2023].

[22] Github Contributors, "pq-crystals/kyber," *github.com*, Jun. 27, 2017. [Online]. Available: https://github.com/pq-crystals/kyber. [Accessed: Mar. 1, 2023].

[23] P. Schwabe, "Kyber – Software," *pq-crystals.org*, Feb. 3, 2023. [Online]. Available: https://pq-crystals.org/kyber/software.shtml. [Accessed: Mar. 1, 2023].

[24] I. Grigorik, "Networking 101: Transport Layer Security (TLS)," *hpbn.co*, 2013. [Online]. Available: https://hpbn.co/transport-layer-security-tls/. [Accessed: Mar. 2, 2023].

[25] Wikipedia Contributors, "Transport Layer Security," *wikipedia.org*, Dec. 7, 2001. [Online]. Available: https://en.wikipedia.org/wiki/Transport_Layer_Security. [Accessed: Mar. 2, 2023].

[26] Domantus G., "What is SSH: Understanding Encryption, Ports and Connection," *hostinger.com*, Mar. 3, 2023. [Online]. Available: https://www.hostinger.com/tutorials/ssh-tutorial-how-does-ssh-work. [Accessed: Mar. 2, 2023].

[27] D. Stebila and J. Green, "Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer," IETF RFC 5656, Dec., 2009. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5656. [Accessed: Mar. 2, 2023].

[28] D. B. Johnson, "Post-quantum algorithm vulnerable to side channel attacks," *scmagazine.com*, Feb. 22, 2023. [Online]. Available: https://www.scmagazine.com/analysis/policy/post-quantum-algorithm-attack. [Accessed: Mar. 3, 2023].

[29] L. Botros, M. Kannwischer, and P. Schwabe, “Memory-Efficient High-Speed Implementation of Kyber on Cortex-M4,” in *Progress in Cryptology – Africacrypt 2019*, 2019, vol. 11627, pp. 209-228.

[30] H. Fujii and D. Aranha, “Curve25519 for the Cortex-M4 and Beyond,” in *Progress in Cryptology – Latincrypt 2017*, 2017, vol. 11368, pp. 109-127.

[31] N. Sullivan, “Securing the post-quantum world,” *cloudflare.com*, Dec. 11, 2020. [Online]. Available: https://blog.cloudflare.com/securing-the-post-quantum-world/. [Accessed: Mar. 5, 2023].

[32] A. Weibel, “Round 2 post-quantum TLS is now supported in AWS KMS,” *amazon.com*, Nov. 16, 2020. [Online]. Available: https://aws.amazon.com/blogs/security/round-2-post-quantum-tls-is-now-supported-in-aws-kms/. [Accessed: Mar. 5, 2023].

[33] M. Lantz and M. Hill, “World’s First Quantum Computing Safe Tape Drive,” *ibm.com*, Aug. 23, 2019. [Online]. Available: https://www.ibm.com/blogs/research/2019/08/crystals/. [Accessed: Mar. 5, 2023].

[34] L. Wan, F. Zheng, G. Fan, R. Wei, L. Gao, Y. Wang, J. Lin, and J. Dong, “A Novel High-performance Implementation of CRYSTALS-Kyber with AI Accelerator,” in *Computer Security – ESORICS 2022*, 2022, vol. 13556, pp. 514-534.

[35] T. Kamucheka, A. Nelson, D. Andrews and M. Huang, "A Masked Pure-Hardware Implementation of Kyber Cryptographic Algorithm," in *2022 International Conference on Field-Programmable Technology (ICFPT)*, 2022, pp. 1-1, doi: 10.1109/ICFPT56656.2022.9974404.