Performance Evaluation of Quantum-Secure Symmetric Key Agreement

Amin Rois Sinung Nugroho
Macquarie University
aminroissinung.nugroho@hdr.mq.edu.au

Muhammad Ikram
Macquarie University
muhammad.ikram@mq.edu.au

Mohamed Ali Kaafar
Macquarie University
dali.kaafar@mq.edu.au

***Abstract*—Quantum-safe public key exchange protocols face significant challenges in hardware- and software-based approaches. Quantum key distribution, which relies on specialized quantum hardware, presents a significant barrier to widespread adoption due to its high cost and limited scalability. Conversely, software-based solutions using post-quantum algorithms introduce complications, such as increased resource demands and larger ciphertexts. Furthermore, the security of these post-quantum algorithms remains relatively untested, which has led to the emerging trend of hybrid deployment, combining classical and quantum-resistant techniques to hedge against potential vulnerabilities.**

Recently, Arqit proposed a quantum-secure symmetric key agreement (SKA) protocol, claiming that it is lightweight and scalable [1] to address these problems. However, their proprietary solution is not available for independent analysis. To evaluate the performance and scalability of quantum-secure SKA techniques, we develop variations of the SKA protocol using open-source and accessible components in this work.

To analyze quantum-secure SKA scheme, we implemented an SKA technique that involves a hybrid mechanism, leveraging secret strings distributed through a combination of existing classical and quantum public key pairs during the initial key exchange.

We analyze our scheme and demonstrate that it incurs minimal performance overhead, with only 99ms for purely quantum SKA and 199ms for the hybrid version, compared to the classical SKA protocol. We also show that our scheme remains robust under various network conditions, including delays, packet losses, and bandwidth variations, maintaining small and consistent overheads.

We also show that this solution is scalable, with an overhead of only *one* second for every additional five concurrent users. This performance improves significantly with increased computational resources—achieving a 50-60% improvement when scaling from two to four CPUs. Additionally, our security evaluations confirm that the protocol provides consistent and sufficient randomness throughout the key agreement process, ensuring quantum-resistance at every stage.

***Index Terms*—Quantum-Safe Cryptography, Symmetric Key Agreement, Hybrid Encryption, Quantum-Resistant Algorithms, Network Security**

I. Introduction

Quantum computing presents a significant challenge to the security infrastructure of modern cryptographic systems. While the full-scale deployment of quantum computers may seem distant, the potential for "harvest now, decrypt later" attacks [2], [3], [4]–where adversaries collect encrypted data now with the intent of decrypting it once quantum capabilities mature–is a growing concern. This scenario underscores the urgency of developing quantum-resistant cryptographic solutions to secure data against future quantum adversaries.

Most of the leading candidates [5], [6], [7] from the NIST post-quantum cryptography competition, including lattice-based algorithms, rest on theoretical problems [8] that remain poorly understood outside niche academic circles. This limited understanding raises the possibility that undiscovered vulnerabilities [3], [9], [4] could be exploited in the future, undermining the very foundation of these algorithms. Additionally, these algorithms often demand significantly more computational resources and larger ciphertext sizes, introducing practical challenges for their *performance evaluation and, thus, widespread deployment*.

This uncertainty has driven the adoption of hybrid cryptographic approaches that combine classical and quantum-resistant algorithms to provide a dual layer of protection. A hybrid scheme ensures that, even if vulnerabilities in quantum-safe algorithms are discovered later, the system remains secure, as an attacker would need to break both the classical and quantum-resistant layers simultaneously. While this hybrid approach offers enhanced security, it also introduces added complexity and overhead, particularly in resource-constrained environments.

To address the challenges of secure key exchange in the quantum era, Arqit recently proposed a quantum-secure symmetric key agreement (SKA) protocol, emphasizing its lightweight and scalable characteristics [1]. However, as their solution is proprietary, its performance and scalability remain opaque. In this work, we develop a system with a variations of the SKA technique using open-source and accessible components to evaluate the performance and scalability of quantum-secure SKA protocols.

Our work main contributions are:

- We present an open implementation of a hybrid

quantum SKA protocol that optimally balances security, performance, and scalability.
- We demonstrate that our system with the SKA protocols introduces minimal overhead compared to classical SKA, with only 99ms for the quantum version and 199ms for the hybrid version. These small overheads remain consistent even under varying network conditions, including simulated delays, packet losses, and bandwidth limitations. Our SKA implementation also scales efficiently, with only around 1 second of additional overhead for every 5 concurrent users. Moreover, increasing the CPU count from 2 to 4 results in a 50-60% performance improvement, highlighting the protocol's adaptability to increased computational resources.
- Our security evaluations confirm that this hybrid SKA protocol ensures high levels of randomness throughout the key agreement process. We observed that the keys and ciphertexts generated by quantum-safe algorithms exhibit higher entropy to countermeasure entropy-based profiling attacks on encrypted network traffic.

The remainder of this paper is organized as follows: Section II provides a detailed review of the current developments in quantum-safe key distribution. Section III formally introduces our implemented SKA scheme. Section IV describes the experimental setup used to evaluate the performance of our solution and presents the results of our performance evaluations, while Section V concludes the paper.

II. Background and Related Work

To respond to the aforementioned quantum computing challenges, the National Institute of Standards and Technology (NIST) launched a global initiative in 2016 to standardize quantum-safe cryptographic algorithms, thereby protecting data from future quantum attacks [10]. NIST's effort, structured as an open global competition, invited research teams to propose quantum-safe cryptographic schemes. Each submission had to provide detailed specifications and implementation codes for algorithms designed either for key exchange mechanisms or digital signatures. Following three rounds of stringent evaluations, NIST identified a shortlist of finalist algorithms to be standardized in the near future. For public-key encryption and key exchange, NIST has selected ML-KEM (Module-Lattice based Key Encapsulation Mechanism), previously known as CRYSTALS-KYBER, as the finalist for Key Encapsulation Mechanism (KEM) [5]. On the other hand, in the realm of digital signatures, three algorithms have been chosen as finalists. The ML-DSA (Module-Lattice based Digital Signature Algorithm), formerly CRYSTALS-Dilithium, represents a leading solution [7]. Another finalist, SLH-DSA (Stateless Hash-Based Digital Signature Algorithm), is based on the SPHINCS+ framework [11], while Falcon (Fast Fourier Lattice-based Compact Signatures Over NTRU Lattices) rounds out the selection [6]. These algorithms are poised to safeguard the cryptographic foundations of modern computing in the face of quantum advancements.

While lattice-based cryptography has emerged as a favored approach due to its theoretical resistance to quantum attacks, the reliance on structured lattice problems also raises concerns. The complexity of these problems, coupled with the relative novelty of the algorithms, means that future vulnerabilities could be discovered. This uncertainty has led NIST to keep several non-lattice-based algorithms, such as BIKE and Classic McEliece, in the competition for further evaluation [12], [13]. The ongoing post-quantum cryptography standardization process remains dynamic, with the cryptographic community continuing to assess the resilience and practicality of these emerging algorithms.

Hybrid SKA schemes. Despite the progress in developing quantum-safe algorithms, two key concerns continue to preoccupy the research community. First, the foundation of most NIST finalist algorithms, the lattice problem, remains complex and not fully understood by a broad segment of the cryptographic community. This complexity raises concerns that unforeseen vulnerabilities may be discovered in the future. Second, these algorithms are relatively new and have not undergone extensive, long-term testing, making their resilience uncertain over time. Consequently, NIST is encouraging proposals based on non-lattice problems to diversify the cryptographic landscape. The inclusion of non-lattice-based algorithms could prove crucial if vulnerabilities in lattice-based methods are identified at a later stage.

Given these concerns, a hybrid deployment trend has emerged, wherein systems integrate both classical and quantum-safe cryptographic methods. This hybrid approach provides an additional layer of security, ensuring that even if quantum-safe algorithms are compromised, the classical algorithms can still offer protection. In such systems, an attacker would need to break both the classical and quantum-safe cryptographic schemes simultaneously, which is currently infeasible, thereby maintaining security comparable to traditional cryptography.

Most recently, several major technology firms have already begun deploying hybrid quantum-safe ciphers in production environments. For instance, Apple has integrated quantum-safe cryptography into its iMessage platform, which serves nearly two billion active devices. Apple employs a hybrid cipher combining ECDH P256 and quantum-safe Kyber, reinforcing the platform's security against quantum threats [9]. Amazon Web Services (AWS) has similarly enabled quantum-safe options for

its file transfer services through AWS Transfer Family, allowing users to select from ECDH curves such as P256 or X2551 to combine with Kyber for secure file transfers over OpenSSH [4]. Additionally, both Google and Cloudflare have implemented hybrid cryptography in their services, utilizing a combination of the classical ECDH X25519 and the quantum-safe Kyber cipher in parts of their production infrastructure [14], [3].

Performance evaluation. To migrate to quantum-safe cryptographic systems, it is important to evaluate the trade-offs involved, including security, usability, computational overhead, and implementation complexity. Quantum-safe algorithms, particularly those based on structured lattices, typically require larger key sizes and ciphertexts, resulting in greater resource consumption. Evaluating the performance of these algorithms in widely used protocols, such as TLS and SSH, is a critical step toward practical deployment [15].

Several studies have assessed the performance of post-quantum cryptographic algorithms under real-world conditions. For instance, Sikeridis et al. evaluated the overhead of post-quantum key encapsulation mechanisms and authentication mechanisms within TLS 1.3 and SSH, demonstrating that the additional computational costs of these algorithms were generally manageable. Their study involved extensive testing over various network conditions and concluded that the performance impact could be mitigated through techniques such as adjusting the TCP window size [16], [17]. Similar findings were reported by Google and Cloudflare during their quantum-safe TLS experiments, which tested various post-quantum algorithms under diverse network configurations [18].

In contrast, another benchmarking approach utilized emulated virtual networking within the Linux kernel. This method offers a cost-effective alternative, as it can be conducted on a single machine while simulating specific network conditions such as latency and packet loss. Paquin *et al.* employed this approach for post-quantum TLS benchmarking, allowing researchers to *explore network performance under more controlled conditions* [19]. Kampanakis *et al.* analyzed the data-heavy TLS 1.3 sessions, emphasizing the Time-To-Last-Byte (TTLB) metric–representing the total duration from the initiation to the conclusion of a TLS connection—rather than focusing solely on handshake performance [20]. These diverse measurement techniques provide a baseline for the performance evaluations of our implemented SKA scheme.

Security evaluation of quantum-safe SKAs. A fundamental aspect of cryptographic security is the randomness of key generation. Insufficient randomness can lead to predictable keys, allowing adversaries to infer patterns from encrypted messages. To address this, we will evaluate the randomness properties of cryptographic keys generated by post-quantum schemes, adhering to the guidelines set forth by OWASP and MITRE, which define best practices for ensuring sufficient randomness [21], [22]. Corrigan-Gibbs *et al.* have introduced a framework that systematically assesses key randomness by integrating entropy authority, which we will employ as part of our evaluation methodology [23].

In addition to ensuring strong key generation, it is important to evaluate the resilience of post-quantum cryptographic schemes against profiling attacks [24]. Even when encrypted, network traffic can be analyzed and profiled using machine learning techniques. Research has shown that these techniques can identify patterns in encrypted traffic, which can then be used for clustering or fingerprinting, potentially leading to private data leakage or providing attackers with preliminary information for further exploitation [24]. As part of our security evaluation, we will investigate whether post-quantum encrypted traffic is susceptible to such profiling attacks, ensuring that these new cryptographic schemes do not introduce unforeseen vulnerabilities in network security.

To measure the effectiveness of our quantum-safe SKA implementations, we present our implementation of the SKA schemes in the following section.

III. Overview of Our Symmetric Key Agreement Schemes

Key agreement is a fundamental process where two or more parties establish shared cryptographic keys, enabling secure authentication as well as communication. Typically, this process is completed before any sensitive information is exchanged. Previously, this is usually done using public key cryptography. However, as explained in previous chapters, we need to replace classical public key cryptography with quantum-safe versions to prepare against quantum computing threats.

As explained in previous sections, Arqit proposed a symmetric key agreement platform [25] that is quantum-secure, lightweight, and scalable. However, their solution is a propietary product. In this work, we develop variations of SKA schemes for evaluation purposes using open-source and accessible components.

The key novelty of this SKA protocol lies in *minimizing* the performance overhead typically associated with quantum-safe cryptography. While post-quantum algorithms are employed at the beginning of the key agreement process, the subsequent communication relies on lightweight symmetric keys, reducing the computational and network resource burden that post-quantum algorithms would otherwise impose. This design choice ensures that while the key exchange phase is more resource-intensive, the ongoing communication remains

efficient and scalable. We also implement forward secrecy and employ a key management server to facilitate the key exchange process.

In each key agreement scheme, we use TLS authentication which uses RSA [26] encryption algorithm at the very beginning. Then we use either or both elliptic curve cryptography [27] and post-quantum Kyber [5], for the initial secrets encryption, depending on the key agreement scheme.

We also leverage Argon2 [28], a key derivation function, to facilitate key derivation process as part of the key agreement design. In summary, Argon2 has won a password hashing competition in 2015 and has been widely used to derive cryptographic keys from passwords by applying a hash function in a way that makes brute-force attacks and rainbow table attacks computationally expensive and difficult to execute.

Then at the end of our key agreement setting, we use AES-256 to generate the symmetric keys. Advanced Encryption Standard (AES) [29], is the current state of the art of symmetric cryptography which was introduced by NIST in 2001 to replace the outdated Data Encryption Standard (DES). AES with 256 bits key (AES-256) has also been proven to be quantum-secure [30]. More technical details of of each key agreement component will be further explained in Section IV where we describe our experiment setup to evaluate our SKA proposal.

In summary, in our implementation, we use Cosmian, open-source key management software to facilitate the TLS authentication, key generation, key management, and key exchange as Arqit's one is proprietary. In addition, we choose the secret generation method using OpenSSL and the key generation algorithm using ECDH [27] and Kyber [5] as Arqit's ones are not disclosed. We also choose Argon2 as Key Derivation Function where Arqit's hash function is not disclosed. Finally, we define three SKA types (Hybrid, Quantum, Classic) for benchmarking purpose based on encryption algorithms used while Arqit's solution only use post-quantum encryption algorithms.

To formalize the key exchange process, we use mathematical notation to outline the key steps of the protocol. For clarity, we define three cases to represent the various SKA schemes: (A) **Hybrid SKA**–combines classical and post-quantum cryptography, providing dual-layered protection; (B) **Quantum SKA**–utilizes only post-quantum algorithms for key agreement, ensuring quantum resistance; (C) **Classical SKA**–employs traditional cryptographic methods, included in our evaluation for benchmarking purposes.

We formally outline our implementation in Phases 1–10. We begin with an initialization step (Phase 1) and quantum-safe key generation processes (Phase 2) between two entities, A and B, facilitated by a trusted key server (S). The protocol begins with classical authentication through a TLS (Transport Layer Security) handshake, ensuring that both parties are verified before proceeding with key generation.

Depending on the cryptographic scheme in use, the key exchange may follow one of three paths: hybrid (combining post-quantum and classical cryptography), quantum-only, or classical-only. In each case, entities A and B generate secret strings, split them into parts, and encrypt the parts using the appropriate cryptographic scheme (Kyber for post-quantum (Phase 4), ECC for classical (Phase 5), or a combination of both for hybrid (Phase 3)). These encrypted secret parts are then sent to the key server for decryption and key derivation.

Phase 1 : Initialization

1: **Input:** Entities A and B wish to establish a shared symmetric key via a trusted key server S.
2: **Output:** A symmetric key K_s shared between A and B for secure communication.
3: **Step 0:** *Authentication via Classical TLS*
4: A and B authenticate to S using a classical RSA-based TLS certificate. This step ensures that A and B are verified entities:

$$A \xrightarrow{\text{RSA-based TLS}} S, \quad B \xrightarrow{\text{RSA-based TLS}} S$$

Phase 2 : Initial Key Generation

1: **Step 1A:** *Case A: Hybrid Key Generation*
2: A and B request Public-Private Key Pair in post-quantum encryption scheme (Kyber) and classical elliptic curve encryption (ECC) to S.
3: **Step 1B:** *Case B: Quantum Key Generation*
4: A and B request Public-Private Key Pair in post-quantum encryption scheme (Kyber) to S.
5: **Step 1C:** *Case C: Classical Key Generation*
6: A and B request Public-Private Key Pair in classical elliptic curve encryption (ECC) to S.

Phase 3 : Step 2A: *Case A: Hybrid Secret Sharing*

1: Both A and B generate a secret string M_A and M_B, respectively, and split it into four parts. Two parts are encrypted using a post-quantum encryption scheme (Kyber) and two parts using classical elliptic curve encryption (ECC):

$$C_1^A = \text{Kyber}(M_{A,1}), \quad C_2^A = \text{ECC}(M_{A,2})$$
$$C_3^A = \text{Kyber}(M_{A,3}), \quad C_4^A = \text{ECC}(M_{A,4})$$

A sends $\{C_1^A, C_2^A, C_3^A, C_4^A\}$ to S. Similarly, B sends $\{C_1^B, C_2^B, C_3^B, C_4^B\}$ to S.

Phase 4 Step 2B: *Case B: Quantum Secret Sharing*

1: Both A and B generate a secret string M_A and M_B, respectively, and split it into four parts encrypted using a post-quantum encryption scheme (Kyber):

$$C_1^A = \text{Kyber}(M_{A,1}), \quad C_2^A = \text{Kyber}(M_{A,2})$$
$$C_3^A = \text{Kyber}(M_{A,3}), \quad C_4^A = \text{Kyber}(M_{A,4})$$

A sends $\{C_1^A, C_2^A, C_3^A, C_4^A\}$ to S. Similarly, B sends $\{C_1^B, C_2^B, C_3^B, C_4^B\}$ to S.

Phase 5 Step 2C: *Case C: Classical Secret Sharing*

1: Both A and B generate a secret string M_A and M_B, respectively, and split it into four parts encrypted using classical elliptic curve encryption (ECC):

$$C_1^A = \text{ECC}(M_{A,1}), \quad C_2^A = \text{ECC}(M_{A,2})$$
$$C_3^A = \text{ECC}(M_{A,3}), \quad C_4^A = \text{ECC}(M_{A,4})$$

A sends $\{C_1^A, C_2^A, C_3^A, C_4^A\}$ to S. Similarly, B sends $\{C_1^B, C_2^B, C_3^B, C_4^B\}$ to S.

The key server, S, decrypts the received secret shares, depending on the scheme, either using Kyber (for quantum encryption–Phase 7), ECC (for classical encryption–Phase 8), or both (in the hybrid case–Phase 6). After decryption, the server combines (cf. Phase 9) the decrypted secret strings from both A and B, hashes them to derive a shared key, K_{AB}. This key is then used to encrypt a trusted symmetric key K_s, which the server sends back to A and B. For each new session, a unique session key is derived (cf. Phase 10) by hashing K_s with the session index, ensuring forward secrecy and secure communication for ongoing sessions. The use of this SKA protocol balances performance efficiency with the robustness required for future quantum-safe communication.

IV. Experiment Setup and Result Evaluation

To evaluate the performance of our implemented SKA schemes, we designed a comprehensive test-bed with configurations and measurement goals. The test-bed setup, depicted in Figure 1, enables us to explore different deployment scenarios, such as key generation under constrained resources and the impact of scaling on performance. Our test-bed consists of four virtual machines (VMs) with different specifications to assess both the *scalability* and *efficiency* of the SKA protocols under varying resource constraints. Our initial experiments utilize AWS `T3.Small` instances (2 CPUs, 2 GB RAM, Ubuntu 22.04 LTS) simulate a typical deployment environment with limited resources. To examine the effect of scaling computational power, we also employ

Phase 6 Step 3A: Case A: Hybrid Decryption and Key Derivation

1: The key server S decrypts the received parts using the corresponding decryption algorithms for Kyber and ECC:

$$M_A = \text{Kyber}^{-1}(C_1^A) \parallel \text{ECC}^{-1}(C_2^A) \parallel \text{Kyber}^{-1}(C_3^A) \parallel \text{ECC}^{-1}(C_4^A)$$

$$M_B = \text{Kyber}^{-1}(C_1^B) \parallel \text{ECC}^{-1}(C_2^B) \parallel \text{Kyber}^{-1}(C_3^B) \parallel \text{ECC}^{-1}(C_4^B)$$

After decryption, S combines the secrets from both users, hashes them, and derives a shared key K_{AB}:

$$K_{AB} = H(M_A \parallel M_B)$$

Phase 7 Step 3B: Case B: Quantum Decryption and Key Derivation

1: The key server S decrypts the received parts using the corresponding decryption algorithms for Kyber:

$$M_A = \text{Kyber}^{-1}(C_1^A) \parallel \text{Kyber}^{-1}(C_2^A) \parallel \text{Kyber}^{-1}(C_3^A) \parallel \text{Kyber}^{-1}(C_4^A)$$

$$M_B = \text{Kyber}^{-1}(C_1^B) \parallel \text{Kyber}^{-1}(C_2^B) \parallel \text{Kyber}^{-1}(C_3^B) \parallel \text{Kyber}^{-1}(C_4^B)$$

After decryption, S combines the secrets from both users, hashes them, and derives a shared key K_{AB}:

$$K_{AB} = H(M_A \parallel M_B)$$

AWS `C5.XLarge` instances (4 CPUs, 8 GB RAM, Ubuntu 22.04 LTS), allowing us to measure how increasing the CPU count impacts performance. This two-tiered setup enables us to assess the SKA protocol's base performance and scalability comprehensively.

Our experimental setup replicates a realistic network environment, with one VM acting as the key management server (KMS), S (cf. § III), running the Cosmian key management software [31]–an open-source solution chosen for its support of advanced cryptographic protocols. Two additional VMs are designated as benign entities (cf. § III), A and B, who engage in key agreement processes, while the fourth VM, M, simulates a Man-in-the-Middle (MiTM) attacker to test the security and robustness of the protocol under adversarial conditions. The objective of these experiments is twofold: *first*, to quantify the performance overhead of the SKA protocol in terms of elapsed time and CPU utilization, and

Phase 8 Step 3C: Case C: Classical Decryption and Key Derivation

1: The key server S decrypts the received parts using the corresponding decryption algorithms for ECC:

$$M_A = \text{ECC}^{-1}(C_1^A) \parallel \text{ECC}^{-1}(C_2^A) \parallel \text{ECC}^{-1}(C_3^A) \parallel \text{ECC}^{-1}(C_4^A)$$

$$M_B = \text{ECC}^{-1}(C_1^B) \parallel \text{ECC}^{-1}(C_2^B) \parallel \text{ECC}^{-1}(C_3^B) \parallel \text{ECC}^{-1}(C_4^B)$$

After decryption, S combines the secrets from both users, hashes them, and derives a shared key K_{AB}:

$$K_{AB} = H(M_A \parallel M_B)$$

Phase 9 Symmetric Key Generation and Delivery

1: **Step 4:** *Symmetric Key Generation and Delivery*
2: The server S generates an initial trusted symmetric key K_s and encrypts it with the derived key K_{AB}:

$$C_{K_s} = \text{Enc}_{K_{AB}}(K_s)$$

The server S sends C_{K_s} to both A and B for initial authentication.

second, to evaluate its scalability when multiple users concurrently establish key agreements with the KMS.

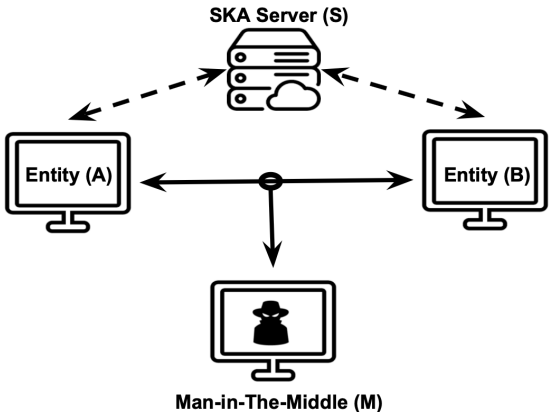


Fig. 1: Our evaluation testbed for the SKA schemes. Here, we have two entities, A and B, that perform our SKA by exchanging (shown in dotted-arrows) encrypted secrets authenticated by the key management server, S. Once A and B are authenticated, they can communicate (shown in solid arrows). However, entity M can act as an MiTM attacker who can eavesdrop, store, or profile the (encrypted) traffic between A and B.

To ensure the accuracy and reliability of our measurements, we collect elapsed time and CPU utilization

Phase 10 Session Key Derivation

1: **Step 5:** *Session Key Derivation*
2: For each new session, the initial symmetric key K_s is hashed to derive a unique session key $K_{s,i}$:

$$K_{s,i} = H(K_s \parallel i)$$

This ensures forward secrecy by generating a new key for every session i.

metrics using GNU Time [32], repeating each experiment 30 times to calculate the average and mitigate statistical variance. This repetition ensures that any network fluctuations or system anomalies are accounted for, allowing for robust conclusions about the protocol's performance. The SKA protocols (quantum, hybrid, and classical) are tested across various configurations, simulating different network conditions and user loads.

In addition to performance metrics, we address security concerns by configuring the key server as a certificate authority (CA) to generate TLS certificates for user authentication, as described in Section III. This ensures that communication between entities A, B and the KMS (S) is authenticated using secure classical algorithms. To prevent unauthorized access, we also employ Ubuntu's built-in firewall to restrict connections to only trusted hosts. Finally, to streamline the experimentation process, we automate our test scenarios using bash scripts that execute relevant Ubuntu commands and interact with the Cosmian API, following the SKA phases outlined in Section III. This automation guarantees consistency across experiments and facilitates large-scale testing to validate our SKA protocol under real-world conditions.

Measuring the performance of our implementation of SKA scheme in simulated network settings. We measure the total elapsed time required to complete the entire SKA process and compare the performance overhead across different SKA types (classical, quantum, and hybrid). This initial comparison is conducted under default network conditions in our AWS environment, which provides a bandwidth of 4 Gbps with zero delay and zero packet loss. These ideal conditions represent a high-performance network environment, offering a baseline for performance metrics.

However, real-world networks often experience suboptimal conditions, such as packet loss, varying latencies, and bandwidth constraints. To measure the impact in these potential deployment challenges, we introduce network emulation techniques using the Linux kernel's network emulation tools to simulate more realistic environments [19], [20]. By varying packet loss rates– 5%, 10%, 15%, and 20%—we model different network qualities. Low packet loss reflects high-quality wired networks, while moderate to high loss simulates

unreliable or congested networks, such as overloaded servers or congested Wi-Fi environments. Additionally, we assess the effect of geographic distances by introducing artificial network delays of 30 ms, 70 ms, and 200 ms, corresponding to close, medium, and far distances between communicating parties. Finally, we evaluate how varying bandwidth capacities (10 Mbps, 50 Mbps, 100 Mbps) affect performance, covering a spectrum from slow to fast network connections. By systematically introducing these network conditions, we aim to capture the protocol's behavior across diverse deployment scenarios, addressing the critical problem of ensuring that the SKA protocol remains performant even in suboptimal environments.

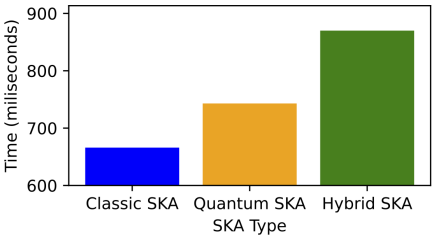
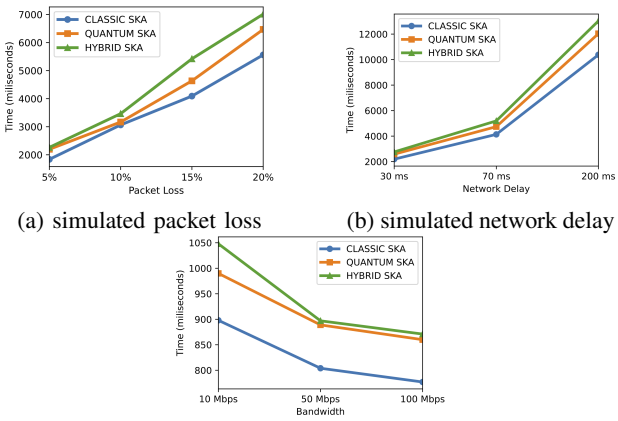


Fig. 2: SKA performance in default network condition.

Figure 2 shows the SKA process elapsed time in default condition (4 Gbps bandwidth between VMs and without any network delay or packet loss). We can observe that compared to classic, our quantum and hybrid SKA(s) have only 99ms (15%) and 199ms (29%) overhead, respectively. Figure 3a shows that compared to Classic SKA, our Quantum and Hybrid SKA(s) showed even smaller overhead between 3% to 32% in percentage and between 103 ms to 1,450 ms in actual value. Figure 3b shows the impact of network delay to simulate geographical distance which is the proportional impact to all SKA schemes with considerable impact in all SKA schemes only at 200 ms delay. Compared to classic SKA, our quantum and hybrid SKA(s) showed a small overhead between 15% to 26% in percentage and between 392 ms to 2,654 ms in actual value for this case. In Figure 3c, our quantum and hybrid SKA(s) showed overhead between 10% to 16% in percentage and between 83 ms to 151 ms in actual value compared to classic SKA in different bandwidth capacities.

Performance measurement for SKA sub-processes. To gain a deeper understanding of the performance characteristics of our SKA protocol, we break down the SKA operation into its individual sub-process and measure the overhead associated with each. These sub-processes include public-private key pair generation, encryption, decryption, key derivation, and symmetric key generation. By collecting metrics such as elapsed time and CPU usage for each sub-process, we aim to identify any bottlenecks or settings that contribute disproportionately



(a) simulated packet loss (b) simulated network delay

(c) simulated bandwidth

Fig. 3: SKA performance in various network conditions.

to the overall overhead. This granular analysis allows us to pinpoint specific areas where optimization could improve the protocol's efficiency. Furthermore, evaluating each stage independently enables us to analyze the cost of different cryptographic operations (e.g., post-quantum vs. classical encryption) in a more controlled manner, helping to clarify the trade-offs between security and performance.

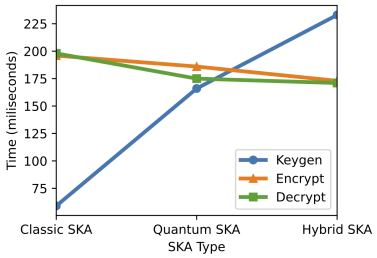
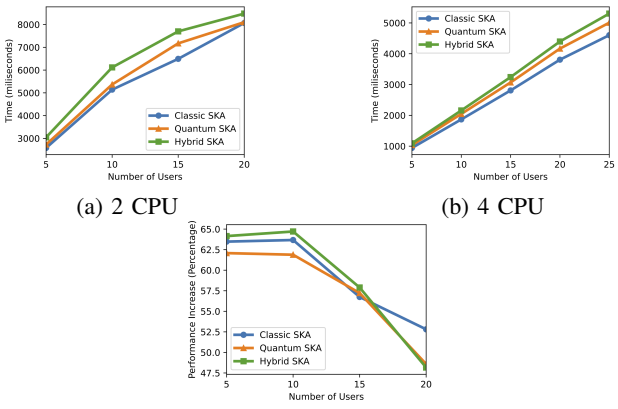


Fig. 4: Comparing performance of key generation, encryption, and decryption between SKA types.



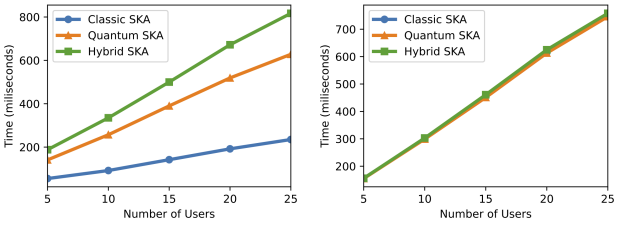
(a) 2 CPU (b) 4 CPU

(c) 4 CPU vs. 2 CPU

Fig. 5: Multi-users SKA performance comparison.

Scalability measurement. Scalability is a critical factor in the deployment of SKA protocols in large-scale environments, where multiple users may need to perform secure key agreements concurrently. To evaluate the scalability of our implemented SKA schemes, we simulate scenarios with increasing numbers of concurrent users–5, 10, 15, 20, and 25 users—running the SKA process simultaneously. This allows us to determine how the system handles increased load and whether the protocol remains efficient as the number of users grows. We will measure both the total elapsed time and the overhead at each stage of the SKA process, identifying how well the system scales in both overall and granular terms. A key goal of this measurement is to assess the protocol's ability to maintain acceptable performance levels as demand increases, ensuring its suitability for real-world, high-traffic applications.

Figure 5 show the results of our experiments. We observe that our SKA solutions are *highly* scalable with only around 1 (one) second overhead per adding 5 concurrent users using only 4 CPU(s). Increasing the CPU(s) from 2 to 4 increases the performance by 50%-60%.

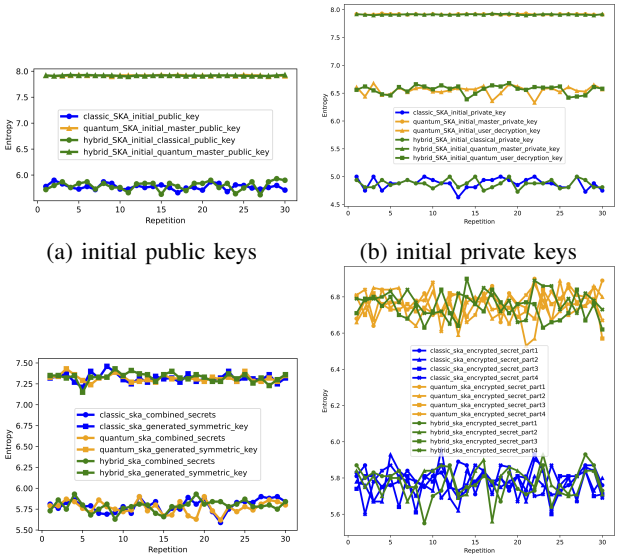


(a) key generation (b) encrypt/decrypt

Fig. 6: Multi-users SKA performance analysis.

Looking closer into the multi-users performance of each SKA stage, most of performance difference between SKA types are coming from the key generation process as can be observed in Figure 6a. Other operations in SKA stages perform similarly between SKA types in a multi-users setting such as the encrypt/decrypt operations as can be observed in Figure 6b.

Randomness of generated secrets and keys. The security of any cryptographic protocol fundamentally depends on the randomness properties of its key generation process. As discussed in Section II, randomness is a critical factor in determining the unpredictability and strength of cryptographic keys. To ensure our SKA protocol meets these requirements, we will evaluate the randomness of all cryptographic materials generated during the SKA process, including public and private keys, encrypted secrets, and derived keys. We quantify randomness by calculating the Shannon entropy of these materials, as high entropy indicates a strong degree of randomness and, therefore, greater security. To achieve statistical validity, we will collect randomness metrics 30 times for each SKA schemes (classical, quantum, and hybrid), allowing us to compare the consistency of the randomness across different cryptographic algorithms. This analysis ensures that our implemented schemes not only performs well but also maintains the highest security standards in terms of key unpredictability.



(a) initial public keys (b) initial private keys

(c) derived keys (d) encrypted secrets

Fig. 7: Entropy measurements of various cryptographic keys and secrets.

We use Shannon entropy to quantify the randomness of all cryptographic materials in our SKA schemes. Using a base 2 logarithm, the maximum possible value of Shannon entropy for 256 bits is 8. Figure 7 demonstrates that our scheme maintains consistency and sufficient randomness throughout all stages of the SKA process, with values approaching the maximum possible Shannon entropy. We confirm that our SKA schemes consistently achieves near-maximum Shannon entropy, ensuring robust randomness.

We observed that keys and ciphertexts generated by quantum-safe algorithms exhibit higher entropy (randomness) values compared to classical algorithms. We determine if the higher entropy values of quantum-safe encrypted materials can be leveraged to classify encrypted network traffic. As a future work, we aim to investigate the potential of using entropy values to classify encrypted network traffic. If successful, this could enable attackers to identify and target classical encrypted traffic for their "harvest now, decrypt later" strategies.

V. Concluding Remarks

We presented an open implementation of hybrid and quantum-secure SKA as a solution for key exchange

protocol capable of withstanding quantum attacks while remaining lightweight, robust, and scalable. Our performance evaluations indicated that our quantum and hybrid SKA implementations introduce minimal overheads of 99ms and 199ms, respectively, compared to classical SKA. These overheads remain consistent across various simulated network conditions, including delays, packet losses, and bandwidth variations. Our SKA implementations demonstrate high scalability, with an additional overhead of only one second for every five concurrent users when utilizing four CPUs. Furthermore, increasing the number of CPUs from two to four results in a performance boost of 50%-60%. Our security evaluations reveal that keys and ciphertexts generated by quantum-safe algorithms exhibit higher entropy values, indicating greater randomness. These findings highlight the viability of our hybrid and quantum SKA solutions in providing secure, efficient, and scalable key exchange protocols in the face of emerging quantum threats.

As a future work, we plan to explore the potential of using entropy values to classify encrypted network traffic, which could assist in identifying classical encrypted traffic for "harvest now, decrypt later" attacks. Additionally, we aim to conduct a comprehensive security analysis using various attacker models on the SKA schemes listed in Section III.

REFERENCES

[1] Arqit, "The arqit symmetric key agreement platform whitepaper," May 2024. [Online]. Available: https://arqit.uk/whitepapers/ska-platform-tm-the-arqit-symmetric-key-agreement-platform

[2] Harvest now, decrypt later. [Online]. Available: https://en.wikipedia.org/wiki/Harvest_now,_decrypt_later

[3] Google. Google threat model for post quantum cryptography. [Online]. Available: https://bughunters.google.com/blog/5108747984306176/google-s-threat-model-for-post-quantum-cryptography

[4] Amazon Web Services. Post-quantum hybrid sftp file transfers using aws transfer family. [Online]. Available: https://aws.amazon.com/blogs/security/post-quantum-hybrid-sftp-file-transfers-using-aws-transfer-family/

[5] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, and D. Stehlé, "CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018, pp. 353–367, http://cryptojedi.org/papers/\#kyber.

[6] Pierre-Alain Fouque et al. Falcon. [Online]. Available: https://falcon-sign.info/

[7] Léo Ducas et al. Crystals dilithium. [Online]. Available: https://pq-crystals.org/dilithium/index.shtml

[8] Lattice based cryptography. [Online]. Available: https://en.wikipedia.org/wiki/Lattice-based_cryptography

[9] "imessage with pq3: The new state of the art in quantum-secure messaging at scale," Feb 2024. [Online]. Available: https://security.apple.com/blog/imessage-pq3/

[10] NIST. Post-quantum cryptography. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography

[11] Andreas Hülsing et al. Sphincs+. [Online]. Available: https://sphincs.org/index.html

[12] Nicolas Aragon et al. Bike - bit flipping key encapsulation. [Online]. Available: https://bikesuite.org/

[13] Daniel J. Bernstein et al. Classic mceliece. [Online]. Available: https://classic.mceliece.org/

[14] Cloudflare. The tls post quantum experiment. [Online]. Available: https://blog.cloudflare.com/the-tls-post-quantum-experiment/

[15] Salman, Tara, "Performance analysis of traditional cryptosystems in multi-cloud management platform." [Online]. Available: https://www.cse.wustl.edu/\%7Ejain/cse567-17/ftp/adn_sec.pdf

[16] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Post-quantum authentication in tls 1.3: a performance study," *Cryptology ePrint Archive*, 2020. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/post-quantum-authentication-in-tls-1-3-a-performance-study/

[17] ——, "Assessing the overhead of post-quantum cryptography in tls 1.3 and ssh," in *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 149–156. [Online]. Available: https://doi.org/10.1145/3386367.3431305

[18] Cloudflare. Sizing up post quantum signatures. [Online]. Available: https://blog.cloudflare.com/sizing-up-post-quantum-signatures

[19] C. Paquin, D. Stebila, and G. Tamvada, "Benchmarking post-quantum cryptography in tls," in *Post-Quantum Cryptography: 11th International Conference, PQCrypto 2020, Paris, France, April 15–17, 2020, Proceedings 11*. Springer, 2020, pp. 72–91.

[20] P. Kampanakis and W. Childs-Klein, "The impact of data-heavy, post-quantum tls 1.3 on the time-to-last-byte of real-world connections," Cryptology ePrint Archive, Paper 2024/176, 2024, https://eprint.iacr.org/2024/176. [Online]. Available: https://eprint.iacr.org/2024/176

[21] OWASP. A02:2021 – cryptographic failures. [Online]. Available: https://owasp.org/Top10/A02_2021-Cryptographic_Failures/

[22] Mitre. Cwe-331: Insufficient entropy. [Online]. Available: https://cwe.mitre.org/data/definitions/331.html

[23] H. Corrigan-Gibbs, W. Mu, D. Boneh, and B. Ford, "Ensuring high-quality randomness in cryptographic key generation," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 685–696. [Online]. Available: https://doi.org/10.1145/2508859.2516680

[24] E. Papadogiannaki and S. Ioannidis, "A survey on encrypted network traffic analysis applications, techniques, and countermeasures," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3457904

[25] Arqit, "The arqit symmetric key agreement platform," https://arqit.uk/arqit-ska-platform, 2024, accessed: 2024-11-21.

[26] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[27] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[28] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2: The memory-hard function for password hashing and other applications," in *Proceedings of the 2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 292–302.

[29] National Institute of Standards and Technology (NIST), "Announcing the advanced encryption standard (aes)," National Institute of Standards and Technology, Gaithersburg, MD, USA, Tech. Rep. FIPS PUB 197, November 2001. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[30] X. Bonnetain, M. Naya-Plasencia, and A. Schrottenloher, "Quantum security analysis of aes," *IACR Transactions on Symmetric Cryptology*, vol. 2019, no. 2, pp. 55–93, 2019. [Online]. Available: https://tosc.iacr.org/index.php/ToSC/article/view/8314

[31] Cosmian. Cosmian key management system. [Online]. Available: https://github.com/Cosmian/kms

[32] GNU Project. Gnu time. [Online]. Available: https://www.gnu.org/software/time/