Password authenticated key exchange-based on Kyber for mobile devices

Kubra Seyhan [1] , **Sedat Akleylek** [Corresp., 1, 2, 3] , **Ahmet Faruk Dursun** [1]

[1] Department of Computer Engineering, Ondokuz Mayis University Samsun, Samsun, Turkey

[2] Chair of Security and Theoretical Computer Science, University of Tartu, Tartu, Estonia

[3] Cyber Security and Information Technologies Research and Development Center, Ondokuz Mayis University Samsun, Samsun, Turkey

Corresponding Author: Sedat Akleylek
Email address: akleylek@gmail.com

In this paper, we propose a novel password-authenticated key exchange (PAKE) scheme based on the Crystals-Kyber (Kyber) in the list of selected algorithms: Public-key Encryption and Key-establishment Algorithms of National Institute of Standards and Technology (NIST) Post-Quantum Cryptography project. The main aim of this paper is to construct a PAKE version of Kyber for mobile environments. To add password-based authentication, module learning with errors (MLWE)-based password-authenticated key exchange (PAK) approach, which provides explicit authentication and perfect forward secrecy, is followed. Since the proposed PAKE also contains Kyber's own authentication, it provides two-way authentication. The constructed Kyber.PAKE is secure against dictionary attacks in the random oracle model (ROM) by using Bellare-Pointcheval-Rogaway (BPR) security model assumptions. According to the implementation results, Kyber.PAKE presents better run-time than lattice-based PAKE schemes with similar features. The CPU cycles are relatively negligible as Kyber.PAKE contains strong security proof. In addition, the implementation results for mobile devices show that the proposed PAKE scheme can be efficiently used for the post-quantum security of mobile environments.

Password Authenticated Key Exchange-Based on Kyber for Mobile Devices

Kübra Seyhan[1], **Sedat Akleylek**[1,2,3], **and Ahmet Faruk Dursun**[1]

[1]**Department of Computer Engineering, Ondokuz Mayıs University, 55139, Samsun, Turkey**
[2]**Cyber Security and Information Technologies Research and Development Center, Ondokuz Mayıs University, 55139, Samsun, Turkey**
[3]**University of Tartu, Tartu, Estonia**

Corresponding author:
Sedat Akleylek[2]

Email address: akleylek@gmail.com

ABSTRACT

In this paper, we propose a novel password-authenticated key exchange (PAKE) scheme based on the Crystals-Kyber (Kyber) in the list of selected algorithms: Public-key Encryption and Key-establishment Algorithms of National Institute of Standards and Technology (NIST) Post-Quantum Cryptography project. The main aim of this paper is to construct a PAKE version of Kyber for mobile environments. To add password-based authentication, module learning with errors (MLWE)-based password-authenticated key exchange (PAK) approach, which provides explicit authentication and perfect forward secrecy, is followed. Since the proposed PAKE also contains Kyber's own authentication, it provides two-way authentication. The constructed Kyber.PAKE is secure against dictionary attacks in the random oracle model (ROM) by using Bellare-Pointcheval-Rogaway (BPR) security model assumptions. According to the implementation results, Kyber.PAKE presents better run-time than lattice-based PAKE schemes with similar features. The CPU cycles are relatively negligible as Kyber.PAKE contains strong security proof. In addition, the implementation results for mobile devices show that the proposed PAKE scheme can be efficiently used for the post-quantum security of mobile environments.

1 INTRODUCTION

The emergence of the post-quantum era changed the security of conventional public-key cryptosystems (PKC). The traditional PKCs such as key exchange (KE)/key encapsulation mechanism (KEM) and digital signature schemes will be insecure in the presence of large-scale quantum computers with Shor algorithm (Peikert et al., 2016; Akleylek and Seyhan, 2022). In 2016, NIST started a process to set the post-quantum secure standard PKC (NIST, 2022a). In 2022, lattice-based Kyber was determined as the standard in the KEM category. For digital signature usage, lattice-based Crystals-Dilithium, Falcon, and hash-based SPHINCS+ were selected as the standard (NIST, 2022b). Although the standards have been determined, it is still necessary to design and determine cryptosystems that can be used for particular goals and application areas.

One of the applications of PKCs used for specific purposes is PAKE schemes that provide a high-entropy shared key generated using low-entropy password-based authentication. Due to the easy-to-use structure, PAKE schemes do not require special hardware to store high entropy keys (Bellare et al., 2000). The hardness assumptions of these schemes are also based on discrete logarithm and factorization problems like other PKCs. The first PAKE, Encrypted Key Exchange, was proposed by Bellovin and Merritt in 1992 (Bellovin and Merritt, 1992). Many PAKE proposals, including new theoretical models, were presented in the following years (Bellovin and Merritt, 1993; Jablon, 1996; Wu, 1998; Hao and Ryan, 2011; Shin and Kobara, 2012). In addition, IETF, IEEE, and ISO/IEC conducted studies on the standardization of PAKE protocols (Hao and van Oorschot, 2022). The most recent standardization initiative for PAKE schemes was the process initiated by the IETF in 2019. In this call, completed in March 2020, OPAQUE and CPace schemes were declared standard (Hao, 2021). Although the industry has started to prototype PAKE protocols in real applications with these processes, the adaptation of post-quantum secure algorithms is necessary for future security.

With the development of wireless communication technologies, the increasing use of mobile devices has brought the security of these devices into focus. There is a need for post-quantum secure PKCs such as KEM, authenticated key exchange, and PAKE that consider resource limitations for mobile devices (Dabra et al., 2020; Ding et al., 2022). Lattice-based cryptosystems stand out with their strong proof of security, worst-case hardness, efficiency, and post-quantum security features. The number of lattice-based PAKE schemes proposed for the post-quantum security of the mobile environment is quite limited. In (Dabra et al., 2020), an anonymous ring learning with errors (RLWE)-based two-party PAKE for mobile devices was proposed. The security analysis of this scheme, which includes a four-phase approach, was done in the real-or-random (RoR) model. An improved version of (Dabra et al., 2020) with a practical randomized key exchange approach is proposed in (Ding et al., 2022). In (Islam and Basu, 2021), unlike (Dabra et al., 2020; Ding et al., 2022), a three-party four-phase RLWE-based PAKE scheme was constructed for mobile communication. The security analysis was done in the ROM. Although they were not proposed for mobile devices like these schemes, many lattice-based PAKE protocols that include the traditional PAK approach, which provides explicit authentication and perfect forward secrecy (PFS), have also been constructed (Ding et al., 2017; Gao et al., 2017; Liu et al., 2019; Ren et al., 2023; Seyhan and Akleylek, 2023).

1.1 Motivation and Contribution

Ensuring today's and post-quantum security of PAKE protocols, which have uses in credential recovery, device pairing, and end-to-end (E2E) secure channel applications, is one of the open problems in the literature (Ott et al., 2019; Hao and van Oorschot, 2022). Although the strongest candidates are NIST algorithms, PAKE versions of these schemes have not yet been created for mobile devices. The main aim of this paper is to provide a PAKE version of Kyber scheme for the post-quantum era security regarding mobile environment compatibility. The main contributions of this paper to the literature are listed as follows.

- A novel two-party Kyber.PAKE is constructed to meet the post-quantum secure PAKE requirement for general purposes and mobile networks. It is aimed to design the PAKE version of the Kyber algorithm and to examine its mobile environment suitability.
- To propose the PAKE variation of Kyber, the conventional PAK design suite (MacKenzie, 2002) is adapted to MLWE problem since the main security of Kyber is based on MLWE.
- KEM structures and the MLWE-based one-phase PAKE design idea are used simultaneously to construct a PAKE. By combining adapted MLWE-based PAK design and Kyber (Avanzi et al., 2019) structures, a novel two-party Kyber.PAKE is proposed.
- The proposed Kyber.PAKE also provides explicit authentication and PFS without using a trusted third party, public key infrastructure, and signature due to the one-phase PAK structure.
- The security analysis against online dictionary attacks is presented in the ROM by following BPR Bellare et al. (2000) and CDF-Zip models Wang et al. (2017a,b). Since CDF-Zipf characterizes password distribution, theoretical security analysis is performed by better covering the real-world power of the adversary.
- The implementation of the Kyber.PAKE is written in C (Dursun, 2023a) and Java (Dursun, 2023b). The implementation results are presented in cost, CPU cycle, and run-time. Based on Java implementation, the mobile device performance results are also provided.
- According to the comparison analysis, Kyber.PAKE is one of the best choices in terms of performance and applicability for post-quantum secure mobile communication.

1.2 Organization

In Section 2, the notation, basic definitions, and followed security assumptions are given. The proposed Kyber.PAKE and its correctness are defined in Section 3. In Section 4, the detailed security analysis

Table 1. Notations

$\mathbb{Z}_q$: Integers in modulo q.	R^k: k-dimensional vector of polynomials (R).
$\bmod^+$: Let $\alpha \in \mathbb{Z}^+$. $a' = a \bmod^+ \alpha \| a' \in [0,\ldots,\alpha)$.	R_q^k: R^kin $\bmod q$
$\|\|$: Concatenation operator.	κ : Security parameter.
B^ℓ - B^*: Byte array of length ℓ and arbitrary, respectively.	$D_{k,\eta}^{\mathrm{MLWE}}$: MLWE distribution.
$\psi^k_{d\in\{d_t,d_v,d_u\}}$: The correctness distribution over R defined in Remark 1.	B_η: Centered Binomial Distribution (CBD). Let $\eta \in \mathbb{Z}^+$. For $\{(a_i,b_i)\}_{i=1}^{\eta} \leftarrow (\{0,1\}^2)^\eta$, a B_η sample is obtained with $\sum_{i=1}^{\eta}(a_i - b_i)$. If $v \in R$ is chosen by using B_η, for $v \leftarrow^r b_\eta$, the coefficients of v are from distribution B_η. If $v \leftarrow^r R^k$, the coefficients of v are from distribution b_η^k (Bos et al., 2018).
b_η^k: B_η distribution over R^k.	d_t, d_v, d_u: Reconciliation parameters of Kyber.
pw_C : Client's password.	$x \leftarrow^r X$: x is randomly selected from the distribution X.
sid - cid : Server id - Client id. C - S - $V = C \cup S$: Client - Server - Participant Spaces.	$H_1(\cdot) = \mathrm{SHAKE}-128 : \{0,1\}^* \rightarrow R_q^k$.
ε is negligible in κ.	$H_2(\cdot) = \mathrm{SHA3}-256 : \{0,1\}^* \rightarrow \{0,1\}^k$.
$U(\cdot)$: Uniform distribution.	$\bmod^\pm$: Modular reduction. Let $\alpha \in 2\mathbb{Z}^+$. $a' = a \bmod^\pm \alpha \| a' \in (-\alpha/2,\ldots,\alpha/2]$.
$H_3(\cdot) = \mathrm{SHA3}-256 : \{0,1\}^* \rightarrow \{0,1\}^k$ Key derivation function (KDF) is used to obtain k-bit session key.	pk - sk - ssk - ct: Public key - Secret key - Shared secret key - Ciphertext.
$\bmod^+$: Let $\alpha \in \mathbb{Z}^+$. $a' = a \bmod^+ \alpha \| a' \in [0,\ldots,\alpha)$.	$\mathrm{negl}(\kappa)$: Let $\varpi > 0$ and $\kappa > n_0$. If there is an $n_0 \in \mathbb{N}$ such that $\mathrm{negl}(\kappa) < \kappa^{-\varpi}$, negl is called negligible function.
D_{pk}: pk distribution of Kyber KEM defined with $B^{12kn/8+32}$.	D_{ct}: ct distribution of Kyber KEM defined with $B^{d_u kn/8 + d_v n/8}$.
CCA: Chosen-ciphertext attack. PFR: Pseudorandom function.	XOF: Extendable Output Function
NTT: Number-Theoretic Transform NTT^{-1}: Inverse NTT	CPA: Chosen-plaintext attack. PKE: Public Key Encryption

against dictionary attacks is presented. The implementation results are explained in Section 5. Finally, the conclusion is given in Section 6.

2 PRELIMINARIES

The notation is given in Table 1.

2.1 Basic Definitions of Kyber

In the proposed PAKE, the shared key is obtained by using Kyber PKE and KEM functions/components and the password-based authentication is added by following PAK design idea.

Kyber PKE and KEM are recalled in Table 2. To obtain detailed information, we refer to (Avanzi et al., 2019).

In Table 2, KYBER.CCAKEM uses KYBER.CPAPKE functions to obtain key agreement based on MLWE problem. Since the hardness assumption of Kyber and proposed PAKE version are based on MLWE, the key generation is done by following MLWE assumption.

Definition 1 (MLWE (Bos et al., 2018)) *Let $k \in \mathbb{Z}^+$, $a_i \leftarrow^r R_q^k$, $s \leftarrow^r b_\eta^k$, and $e_i \leftarrow^r b_\eta$. MLWE distribution is obtained as follow: $D_{k,\eta}^{MLWE} : (a_i, b_i = a_i^T s + e_i) \in R_q^k \times R_q$.*

The hardness of MLWE is defined by decisional-MLWE (d-MLWE). Let m independent (a_i, b_i) instances are given ($A \in R_q^{m\times k}, b \in R_q^m$). d-MLWE is a problem that decides whether these samples belong to MLWE ($D_{m,k,\eta}^{\mathrm{MLWE}} : (A, b = As + e)$ where $s \leftarrow^r b_\eta^k$ and $e_i \leftarrow^r b_\eta^m$) or uniform distribution ($U(R_q^{m\times k}) \times U(R_q^m)$). Let $\mathbf{A}$ be an adversary to try to solve d-MLWE problem. The advantage (Adv) of $\mathbf{A}$ is defined as follows:

$$\mathrm{Adv}_{m,k,\eta}^{\mathrm{MLWE}}(\mathbf{A}) = \Big| \Pr[b' = 1 : b' \leftarrow \mathbf{A}((A,b) \in D_{m,k,\eta}^{\mathrm{MLWE}})] - \Pr[b' = 1 : b' \leftarrow \mathbf{A}((A,b) \in U(R_q^{m\times k}) \times U(R_q^m))] \Big|$$

In Table 2, some low-order bits which do not affect the correctness probability of decryption are discarded in pk and ct. These functions used to achieve reconciliation and reduce parameters are remembered in Definition 2 (Bos et al., 2018).

Definition 2 (Compress and Decompress Functions (Bos et al., 2018)) *Let $a \in \mathbb{Z}_q$ and $d < \lceil \log_2(q) \rceil$.*
***(i.)** b =Compress$_q(a,d)$: To obtain $\mathbb{Z}_q \rightarrow \{0,\ldots,2^d - 1\}$, Compress is defined as $b = \lceil \frac{2^d}{q} \cdot a \rfloor \bmod^+ 2^d$.*
***(ii.)** b' =Decompress$_q(b,d)$: To obtain $\{0,\ldots,2^d - 1\} \rightarrow \mathbb{Z}_q$, b' is defined as $b' = \lceil \frac{q}{2^d} \cdot b \rfloor$, where b' is an element which is relatively close to b.*

Table 2. Kyber KEM and PKE Structures (Avanzi et al., 2019)

KYBER.CCAKEM.KeyGEN()	**KYBER.CCAKEM.Enc(pk)**	**KYBER.CCAKEM.Dec(c, sk)**
Output: sk$\in$B$^{24kn/8+96}$	**Input:** pk$\in$B$^{12kn/8+32}$	**Input:** $c \in$B$^{d_ukn/8+d_vn/8}$, sk$\in$B$^{24kn/8+96}$
Output: pk$\in$B$^{12kn/8+32}$	**Output:**$c \in$B$^{d_ukn/8+d_vn/8}$, $K \in$B*, where K is ssk	**Output:** $K \in$B*, where K is ssk.
$z \leftarrow$B^{32}	$m \leftarrow$B^{32}	$\mathrm{pk} = \mathrm{sk} + 12 \cdot k \cdot n/8$
$(\mathrm{pk}, \mathrm{sk}') =$KYBER.CPA.PKE.KeyGEN()	$m \leftarrow H(m)$	$\mathrm{h} = \mathrm{sk} + 24 \cdot k \cdot n/8 + 32$
$\mathrm{sk} = (\mathrm{sk}'\|\|\mathrm{pk}\|\|\mathrm{H}(\mathrm{pk})\|\|z)$	$(\bar{K}, r) = G(m\|\|H(\mathrm{pk}))$	$z = \mathrm{sk} + 24 \cdot k \cdot n/8 + 64$
return $(\mathrm{pk}, \mathrm{sk})$	$c =$KYBER.CPAPKE.Enc(pk, m, r)	$m' =$ KYBER.CPAPKE.Dec$(\mathbf{s}, (\mathbf{u}, v))$
	$K = \mathrm{KDF}(\bar{K}\|\|H(c))$	$(\bar{K}', r') = G(m'\|\|h)$
KYBER.CPAPKE.KeyGEN()	**return**(c, K)	$c' =$ KYBER.CPAPKE.Enc(pk, m', r')
Output: sk$\in$B$^{12kn/8}$		**if** $c = c'$ **then return** $K = \mathrm{KDF}(\bar{K}'\|\|H(c))$
Output: pk$\in$B$^{12kn/8+32}$	**KYBER.CPAPKE.Enc**$(\mathbf{pk}, m, r)$	**else return** $K = \mathrm{KDF}(z\|\|H(c))$
$d \leftarrow$B^{32}	**Input:** pk$\in$B$^{12kn/8+32}$, $m \in B^{32}$, $r \in$B^{32}	**return** K
$(\rho, \sigma) = G(d)$	**Output:**$c \in$B$^{d_ukn/8+d_vn/8}$	
$\hat{A} \in R_q^{k \times k}$	$\hat{t} = \mathrm{Decode}_{12}(\mathrm{pk})$	**KYBER.CPAPKE.Dec**$(\mathbf{sk}, c)$
$s, e \in R_q^k(\mathrm{B}_{\eta_1})$	$\rho = \mathrm{pk} + 12 \cdot k \cdot n/8$	**Input:**$c \in$B$^{d_ukn/8+d_vn/8}$, sk$\in$B$^{12kn/8}$
$\hat{s} = NTT(s)$, $\hat{e} = NTT(e)$	$\hat{A} \in R_q^{k \times k}$	**Output:** $m \in B^{32}$
$\hat{t} = \hat{A} \circ \hat{s} + \hat{e}$	$r \in R_q^k(\mathrm{B}_{\eta_1})$	$\mathbf{u} = \mathrm{Decompress}_q(\mathrm{Decode}_{d_u}(c), d_u)$
$\mathrm{pk} = (\mathrm{Encode}_{12}(\hat{t} \mod {}^+q)\|\|\rho)$	$e_1 \in R_q^k(\mathrm{B}_{\eta_2})$	$\mathrm{v} = \mathrm{Decompress}_q(\mathrm{Decode}_{d_v}(c + d_u \cdot k \cdot n/8), d_v)$
$\mathrm{sk} = (\mathrm{Encode}_{12}(\hat{s} \mod {}^+q))$	$e_2 \in R_q(\mathrm{B}_{\eta_2})$	$\hat{\mathbf{s}} = \mathrm{Decode}_{12}(\mathrm{sk})$
return $(\mathrm{pk}, \mathrm{sk})$	$\hat{r} = \mathrm{NTT}(r)$	$m = \mathrm{Encode}_1(\mathrm{Compress}_q(v - NTT^{-1}(\hat{s}^T \circ \mathrm{NTT}(u)), 1))$
	$\mathbf{u} = NTT^{-1}(\hat{A}^T \circ \hat{r}) + e_1$	**return** m
	$v = NTT^{-1}(\hat{t}^T \circ \hat{r}) + e_2 + \mathrm{Decompress}_q(\mathrm{Decode}_1(m), 1)$	
	$c_1 = \mathrm{Encode}_{d_u}(\mathrm{Compress}_q(\mathbf{u}, d_u))$	• XOF is used in key generation
	$c_2 = \mathrm{Encode}_{d_v}(\mathrm{Compress}_q(v, d_v))$	• m: Message, c: Ciphertext
	return $c = (c_1\|\|c_2)$	• H :B$^* \rightarrow$B^{32} and G :B$^{32} \rightarrow$B^{32}

The distribution $|b' - b \mod {}^{\pm}q| \leq \mathrm{B}_q = \lceil q/(2^{d+1}) \rfloor$ is nearly uniform over the integers of maximum magnitude B_q. Note that Definition 2 is defined over $\mathbb{Z}_q$. In Kyber, since $a \in R_q^k$, for each coefficient of a is evaluated in these functions.

Remark 1 *ψ_d^k is obtained as follows. **(i)** Choose a $y \xleftarrow{r} R^k$. **(ii) return** $(y - Decompress_q((Compress_q(y,d)),d)) \mod {}^{\pm}q$ (Bos et al., 2018).*

Although the main operations of Kyber are performed in the NTT domain, all polynomials sent over the channel are in the normal domain. For the transformation of polynomials to be used in the transmission, encode and decode operations are done (Avanzi et al., 2019; Bos et al., 2018).

Definition 3 ***Decode$_\ell$:*** *$B^{32\ell} \to R_q$. It deserializes a 32ℓ bytes array into a polynomial. Let $B^{32\ell}$ is a byte array. Then the output of Decode$_\ell$ is $f = f_0 + f_1X + f_2X^2 + \cdots + f_{255}X^{255}$ with $f_i \in \{0, \ldots, 2^\ell - 1\}$. Encode$_\ell$ is defined as the reverse of Decode$_\ell$.*

The correctness of the proposed Kyber.PAKE is defined by using the correctness assumptions of KYBER.CCAKEM and KYBER.CPAPKE. The main theorems of these schemes are recalled in Theorems 1 and 2, respectively.

Theorem 1 *Let $k \in \mathbb{Z}^+$, $s, e, r, e_1 \leftarrow b_\eta^k$, $e_2 \leftarrow b_\eta$, $c_t \leftarrow \psi_{d_t}^k$, $c_u \leftarrow \psi_{d_u}^k$, $c_v \leftarrow \psi_{d_v}$, and $\delta = Pr[||e^T r + c_t^T r - s^T e_1 - s^T c_u + e_2 + c_v||_\infty \geq \lceil q/4 \rfloor]$. Then, KYBER.CPAPKE scheme runs with $(1-\delta)$ correctness probability (Bos et al., 2018).*

Theorem 2 *Let G be a random oracle (RO) and KYBER.CPAPKE is correct with $(1-\delta)$ probability. KYBER.CCAKEM also runs with $(1-\delta)$ correctness probability (Bos et al., 2018).*

The security analysis of proposed PAKE is done by using the ROM assumptions of Kyber.

Definition 4 (The ROM Security of Kyber KEM (Avanzi et al., 2019)) *Let XOF, H, and G be the ROs, n_{ro} be the maximum number of **A**'s queries to ROs, and B - C be adversaries who have roughly the same run-time as **A**. The Adv of **A** over Kyber KEM in the ROM is defined by Equation* (1).

$$Adv_{\text{Kyber KEM}}^{CCA}(\mathbf{A}) = 2Adv_{k+1,k,\eta}^{MLWE}(B) + Adv_{PRF}^{prf}(C) + 4n_{ro}\delta \tag{1}$$

2.2 Security Model

In the proposed Kyber.PAKE, the password-based authentication is obtained by adapting traditional PAK (MacKenzie, 2002) design to the MLWE problem. The security analysis of this idea is done in the BPR model (Bellare et al., 2000) by showing the resistance against password dictionary attacks. In this section, special terms and definitions of this security model is explained. Let $C \in C$, $S \in S$, $V \in V = C \cup S$, and DS denotes password space which is constructed according to Zipf's rule (Wang et al., 2017b), respectively. In this model, each C has $pw_C \xleftarrow{r} DS$ and related S holds the hash of pw_C. **A** is designed as a probabilistic algorithm, which can control the entire network and provide input for the participant's instances. By using the RO queries, **A** can launch the attacks. Let **S** be a scheme and $\prod_V^i$ be i-th V instance that can only be used once. **A**'s special query band is defined as follows.

• execute(C,i,S,j): **S** occurs between $\prod_C^i$ and $\prod_S^j$. The outputs of executed **S** are sent to **A**.
• send(V,i,M): Message M is sent to $\prod_V^i$. Then, according to **S**, the computations of the scheme are done by $\prod_V^i$. The outputs are sent to **A**.
• reveal(V,i): Let $\prod_V^i$ be an accepted and has its own ssk. As a result of this query, ssk is sent to **A**.
• corrupt(V): It returns the password of V. If $V \in C$, the output is pw_C. Otherwise, $H_1(pw_C)$.
• test(V,i): Let b be the coin of $\prod_V^i$. With this query, **A** tosses b. If $b = 0$, ssk is sent to **A** by $\prod_V^i$. Otherwise, ssk is chosen uniformly at random from ssk space and is returned to **A**.

In the BPR security model, p-id and s-id are the id's of the parties and a session, respectively. n_e, n_s, n_r, n_c, and n_o represent the maximum number of **A**'s execute, send, reveal, corrupt, and RO queries, respectively. Finally, $T_{\exp}$ represents the generation time of the MLWE samples.

According to the BPR analysis, each user can run the scheme multiple times with different partners.

Definition 5 (Instance Partnership (Bellare et al., 2000)) *Let $\prod_U^i$ and $\prod_V^j$ have (p-id$_i$, s-id$_i$, ssk$_i$) and (p-id$_j$, s-id$_j$, ssk$_j$), respectively. If the following conditions are satisfied, $\prod_U^i$ and $\prod_V^j$ are partnered.* ***(i)*** *$U \in C$ and $V \in S$, or $V \in C$ and $U \in S$.* ***(ii)*** *$ssk_i = ssk_j$, p-id$_i$ =V, and p-id$_j$ =U.* ***(iii)*** *s-id$_i$ = s-id$_j$ =s-id, where this value is not null.* ***(iv)*** *A third oracle other than $\prod_U^i$ and $\prod_V^j$ should not have the same s-id.*

In the security analysis, the instance freshness provides PFS.

Definition 6 (Instance Freshness (Bellare et al., 2000; MacKenzie, 2002)) *Let $\prod_W^i$ and $\prod_V^j$ be partner. If none of the following events occurred, $\prod_W^i$ is a fresh instance that provide forward secrecy.* ***(i)*** *a reveal(W,i) query,* ***(ii)*** *a reveal(V,j) query,* ***(iii)*** *a corrupt(V) query before send(W,i,M) and test(W,i) queries.*

By using definitions and query band, the Adv of **A** in the password-based AKE scheme is examined.

Definition 7 (The Adv of an A (Bellare et al., 2000; MacKenzie, 2002)) *Let $\prod_V^i$ be a fresh instance, **S** be the AKE scheme, and $Suc_{AKE}^{\boldsymbol{S}}$ be an event that **A** makes a $b' = test(V,i)$ query. For b that was selected in the test query, if $b' = b$, the Adv of **A** is defined in Equation* (2).

$$Adv_{AKE}^{\boldsymbol{S}}(\boldsymbol{A}) = |2Pr[Suc_{AKE}^{\boldsymbol{S}}] - 1| \tag{2}$$

In the traditional PAK suit, examinations are made considering that the password distribution is similar to the uniform distribution. Since this idea does not cover the real power of adversary, CDF-Zipf is added to characterize the password distribution.

Definition 8 (CDF-Zipf Model (Wang et al., 2017b)) *Let Correctpw be **A**'s event of guessing a correct password with online dictionary attacks, DS be the size of password dictionary, and n_{op} be the maximum number of active online password-guessing attempts by **A** before a corrupt query. The probability of event correctpw in the conventional approaches is $Pr[Correctpw] = \frac{n_{op}}{DS} + negl(\kappa)$. Since these methods underestimate **A**'s power in real-world applications, CDF-Zipf, which provides the characterized password distribution, is preferred to obtain much realistic examination about password guess. Let $C' \in [0.001, 0.1]$ and $f \in [0.15, 0.30]$ be CDF constants that can be computed by linear regression. According to CDF-Zipf, the probability of Correctpw is determined by using Equation* (3).

$$Pr[Correctpw] = C' \cdot n_{op}^{f} + negl(\kappa) \tag{3}$$

3 PROPOSED KYBER.PAKE SCHEME

To obtain the password-authenticated version of Kyber KEM (Avanzi et al., 2019), the one-phase PAK design approach (MacKenzie, 2002), which provides explicit authentication and PFS, is followed. The KYBER.CCAKEM.KeyGen, KYBER.CCAKEM.Enc, and KYBER.CCAKEM.Dec structures, given in Table 2, are used for key generation, encapsulation, and decapsulation. By using these functions, the idea of PAK is added to achieve password-based authentication. Thanks to the MLWE-based PAK and Kyber structures, two-way authentication is obtained. The proposed Kyber.PAKE contains four main sub-phases (C$_0$, S$_0$, C$_1$, and S$_1$) and three flows. The constructed scheme is detailed in Figure 1.

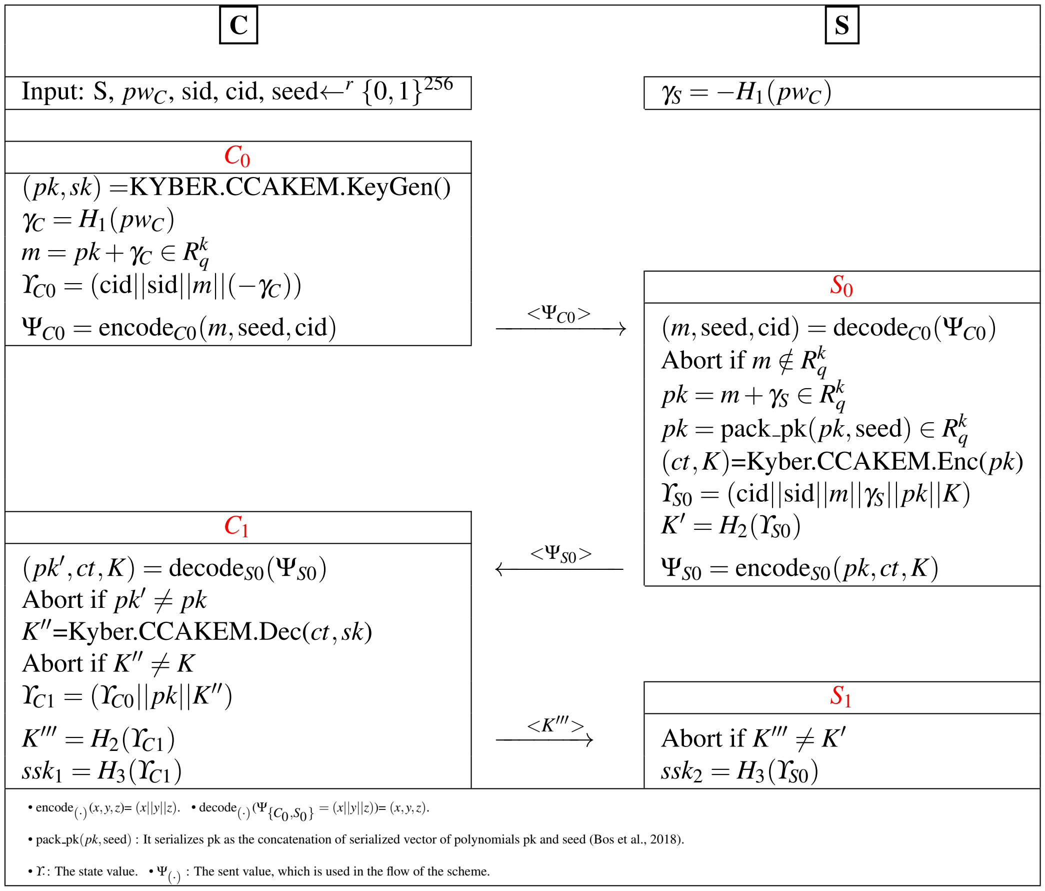
In phase C$_0$ of Figure 1, the key generation is done using the PAK password components (γ_C, γ_S) and KYBER.CCAKEM.KeyGen(). In phase S$_0$, following the PAK idea, **S** obtains **C**'s *pk*, (ct, K) pair with Kyber.CCAKEM.Enc(), and authentication key component (K'). In the C$_1$, authentication checks are done and the key component of **C** is generated with Kyber.CCAKEM.Dec(). For the final authentication check, K''' is generated and the shared key $ssk_1 = H_2(\Upsilon_{C1})$ is computed. In the S$_1$, if $K''' = K'$, $ssk_2 = H_2(\Upsilon_{S0})$ is also generated. At the end of the proposed PAKE, password-authenticated shared secret key $ssk_1 = ssk_2$ is obtained.

3.1 Correctness Analysis of Kyber.PAKE

In Figure 1, if $K = K''$ where (ct, K)=Kyber.CCAKEM.Enc(pk) and K''=Kyber.CCAKEM.Dec(ct, sk), the correctness of Kyber.PAKE is also satisfied. In the Kyber.PAKE, *pk* is computed using the password component. In the S_0 phase of the proposed PAKE, if $pk = m + \gamma_S$ is correctly retrieved by using *m*, there is no changes on the correctness of Kyber.

Let's prove the correctness of Kyber.PAKE based on Theorems 1 and 2.

Figure 1. The Proposed Kyber.PAKE Scheme


C
S
Input: S, pw_C, sid, cid, seed$\xleftarrow{r}\{0,1\}^{256}$
$\gamma_S = -H_1(pw_C)$
C_0
(pk, sk) =KYBER.CCAKEM.KeyGen()
$\gamma_C = H_1(pw_C)$
$m = pk + \gamma_C \in R_q^k$
$\Upsilon_{C0} = (\text{cid}||\text{sid}||m||(-\gamma_C))$
$\Psi_{C0} = \text{encode}_{C0}(m, \text{seed}, \text{cid})$
$<\Psi_{C0}>$
S_0
$(m, \text{seed}, \text{cid}) = \text{decode}_{C0}(\Psi_{C0})$
Abort if $m \notin R_q^k$
$pk = m + \gamma_S \in R_q^k$
$pk = \text{pack\_pk}(pk, \text{seed}) \in R_q^k$
(ct, K)=Kyber.CCAKEM.Enc(pk)
$\Upsilon_{S0} = (\text{cid}||\text{sid}||m||\gamma_S||pk||K)$
$K' = H_2(\Upsilon_{S0})$
$\Psi_{S0} = \text{encode}_{S0}(pk, ct, K)$
$<\Psi_{S0}>$
C_1
$(pk', ct, K) = \text{decode}_{S0}(\Psi_{S0})$
Abort if $pk' \neq pk$
K''=Kyber.CCAKEM.Dec(ct, sk)
Abort if $K'' \neq K$
$\Upsilon_{C1} = (\Upsilon_{C0}||pk||K'')$
$K''' = H_2(\Upsilon_{C1})$
$ssk_1 = H_3(\Upsilon_{C1})$
$<K'''>$
S_1
Abort if $K''' \neq K'$
$ssk_2 = H_3(\Upsilon_{S0})$
• encode$_{(.)}(x,y,z)$= $(x||y||z)$. • decode$_{(.)}(\Psi_{\{C_0,S_0\}} = (x||y||z))$= (x,y,z).
• pack_pk(pk, seed) : It serializes pk as the concatenation of serialized vector of polynomials pk and seed (Bos et al., 2018).
• Υ: The state value. • $\Psi_{(.)}$: The sent value, which is used in the flow of the scheme.

Claim 1 *Let Kyber KEM be correct with* $(1-\delta)$ *probability (Bos et al., 2018). Then, Kyber.PAKE scheme is also correct with* $(1-\delta)$ *probability.*

Proof 1 *According to the detailed definition of and Kyber.CCAKEM.Enc in (Bos et al., 2018), it uses Kyber.CPAPKE.Enc procedure to generate* (ct, K)*, where* $ct = (u, v)$*. In Figure 1, the input of Kyber.CCAKEM.Enc is pk and computed with* $pk = m + \gamma_S$*. Since if the server correctly recover the m from pk with* $pk = m + \gamma_S = pk + \gamma_C + \gamma_S$*, where* $\gamma_C = -\gamma_S$*. By rewriting Remark 1 in (Bos et al., 2018), Equation* (4) *is obtained.*

$$\begin{aligned}
t &= Decompress_q(Compress_q(\overbrace{m}^{pk+\cancel{\gamma_C}} + \cancel{\gamma_S}, d_t), d_t) = As + e + c_t \\
u &= Decompress_q(Compress_q(A^T r + e_1, d_u), d_u) = A^T r + e_1 + c_u \\
v &= Decompress_q(Compress_q(t^T r + e_2 + \lceil q/2 \rfloor \cdot M, d_v), d_v) \\
&= (\overbrace{t}^{As+e+c_t})^T r + e_2 + \lceil q/2 \rfloor \cdot M + c_v \\
&= (As + e)^T r + e_2 + \lceil q/2 \rfloor \cdot M + c_v + c_t^T r \\
&\text{where } c_t, c_u \in R^k,\ c_v \in R
\end{aligned} \tag{4}$$

Since there is no component to change the idea of Remark 1 in (Bos et al., 2018), if $||e^T r + c_t^T r - s^T e_1 - s^T c_u + e_2 + c_v||_\infty \geq \lceil \frac{q}{4} \rfloor$*, then the correctness of Kyber.PAKE is satisfied with* $(1-\delta)$ *probability.*

4 SECURITY ANALYSIS OF KYBER.PAKE

The detailed security analysis is made with PAK suite (MacKenzie, 2002), adapted to MLWE problem. The main aim of the security analysis is to show that the probability of obtaining information about the session key of **A**, who attacks the scheme with an online dictionary attack, is negligible. In the adapted security model, **A** can make the following queries. **(i.)** CA_0: This action is made to instruct $\prod_C^i$ to send the first message to some S. **(ii.)** CA_1: This action is occurred when **A** sends a message to $\prod_C^i$ waiting for the second message of the scheme. **(iii.)** SA_1: This query is done when some messages are sent to $\prod_S^j$. **(iv.)** SA_2: If **A** sends some messages to $\prod_S^j$ waiting for the last message of the scheme, this query is conducted. According to the security analysis, **A** can replace a $\prod_C^i$, a $\prod_S^j$, and partner $\prod_C^i$-$\prod_S^j$ instances by using the mentioned actions and special events, which are given in Table 3.

The Kyber.PAKE's proof of security is conducted by showing that **A** is unable to obtain the new ssk with a more significant Adv than the online dictionary attack. The Adv of **A** is given in Theorem 3.

Theorem 3 *Let the proposed Kyber.PAKE scheme in Figure 1 be represented by* **S**, *DS be the size of the password dictionary,* $|R_q^k| = q^{nk}$, *and the running time of* **A** *be* T. *For* $T' = O(T + (n_o + n_s + n_e)T_{exp})$, *the Adv of* **A** *over the Kyber.PAKE scheme is given in Equation* (5).

$$Adv^{\boldsymbol{S}}_{Kyber.PAKE}(\boldsymbol{A}) \leq O\Big(\frac{(n_e+n_s)(n_e+n_s+n_o)+n_o}{q^{nk}} + \frac{n_s}{2^{\kappa}} + Adv^{CCA}_{Kyber\ KEM}(\boldsymbol{A}) + n_s Adv^{d\text{-}MLWE}_{R_q^k}(T', n_o)\Big) + C' \cdot n_{op}^{f} \tag{5}$$

Proof 2 *Following PAK security analysis (MacKenzie, 2002), schemes* $\mathbf{S0}, \mathbf{S1}, \ldots, \mathbf{S6}$ *where* $\mathbf{S} = \mathbf{S0}$ *are used to prove theorem. In each scheme,* **A** *gains a different feature to make an online dictionary attack. Finally, he/she can create a password guess in the* ***S6***. *The security of the proposed scheme is examined by proving that the Adv of* **A** *obtaining the session key of a fresh instance will be smaller than an online dictionary attack.*

S0*: It is the original Kyber.PAKE scheme.*

S1*: Let m or pk be chosen randomly by honest participants. If these values already appeared in the previous schemes,* ***S1*** *halts and* **A** *fails. In* ***S1****, let* $\varepsilon_1 = \frac{O((n_e+n_s)(n_e+n_s+n_o))}{q^{nk}}$.

Claim 2 *For any* **A**, $Adv^{S0}_{Kyber.PAKE}(\boldsymbol{A}) \leq Adv^{S1}_{Kyber.PAKE}(\boldsymbol{A}) + \varepsilon_1$

Proof 1 *To describe the random selection of m and pk, let's define* $E1$ *and* $E2$. *For* $E = E1 \bigvee E2$, *if the event E occurs, then* ***S1*** *is equal to* ***S0***.

Let $E1$ *be an event defined for* $m = m_1 = m_2 = m_3 = m_4$ *in the following cases:* ***(i)*** *By making* CA_0 *or execute,* m_1 *is obtained.* ***(ii)*** m_2 *is generated by a previous* CA_0 *or execute.* ***(iii)*** m_3 *is used as an input of previous* SA_1. ***(iv)*** m_4 *is utilized in a previous query* $H_{l\in\{2,3\}}(\cdot)$.

Let $E2$ *be an event determined for* $pk = pk_1 = pk_2 = pk_3 = pk_4$ *in the following cases:* ***(i)*** *By making* SA_1 *or execute,* pk_1 *is generated.* ***(ii)*** pk_2 *is obtained by a previous* SA_1 *or execute.* ***(iii)*** pk_3 *is utilized as an input of previous* CA_1. ***(iv)*** pk_4 *is used in a previous query* $H_{l\in\{2,3\}}(\cdot)$.

Considering the events $E1$ *and* $E2$, *it is necessary to examine whether m and pk are previously or newly generated. In these events, the actions* CA_0 *and* SA_1 *are related to send and* $H_{l\in\{2,3\}}(\cdot)$ *queries are associated with RO queries. The previously generated m or pk can be obtained by making send, execute, and RO queries. So, the probability of m or pk occurring in the previous session is* $\frac{(n_e+n_s+n_o)}{|R_q^k|}$. *Since new m or pk can be generated with send and execute, the maximum number of queries is* (n_e+n_s). *Therefore, the probability that E happens is* $\varepsilon_1 = \frac{O((n_e+n_s)(n_e+n_s+n_o))}{q^{nk}}$.

S2*: Unlike* ***S1****, send and execute are replied without answering any RO queries. Afterward, if the RO query is made, the answers are generated as consistently as possible with send and execute. The possible queries and answers in* ***S2*** *are given in Algorithm 1. In* ***S2****, let* $\varepsilon_2 = \frac{O(n_s)}{2^{\kappa}} + \frac{O(n_o)}{|R_q^k|}$.

Claim 3 *For any* **A**, $Adv^{S1}_{Kyber.PAKE}(\boldsymbol{A}) \leq Adv^{S2}_{Kyber.PAKE}(\boldsymbol{A}) + \varepsilon_2$

Table 3. Special Cases of Security Analysis

Event	Input	Output
Testpw(·)	< C,i,S,pw,1 >	For some $\{m, pk, \gamma_S, ct, K\}$, **A** makes; • An $H_l(C,S,m,\gamma_S,pk,K)$ query. • CA$_0$ query for $\prod_C^i$. ⋆ The output is $(m, \text{seed}, \text{cid})$. • CA$_1$ query for $\prod_C^i$. ⋆ The input is (pk, ct, K). • An $H_1(pw)$ query. It returns -γ_S.
	< S,j,C,pw,1 >	For some $\{m, pk, \gamma_S, ct, K\}$, **A** makes; • An $H_l(C,S,m,\gamma_S,pk,K)$ query. • A premade SA$_1$ query is made for $\prod_S^j$. ⋆ The input is $(m, \text{seed}, \text{cid})$. ⋆ The output is (pk, ct, K). • An $H_1(pw)$ query. It returns -γ_S. The associated value of this event is obtained with • The output of $H_{l=\{2,3\}}(\cdot)$ for $\prod_S^j$.
	< C,i,S,j,pw >	For some $l \in \{2,3\}$; Let $\prod_C^i$ and $\prod_S^j$ be mutually partner of each other. • Testpw(S, j, C, pw, l) and Testpw(C, i, S, pw, l) events occur.
Testpw!(·)	< C,i,S,pw >	For some $\{ct, K\}$, • A CA$_1$ query occurs. ⋆ The input is (pk, ct, K). • As a result of CA$_1$, a Testpw$(C, i, S, pw_C, 2)$ occurs.
	< S,j,C,pw >	• A Testpw$(S, j, C, pw, 3)$ event is occured, which is associated with K'''. • By using K''' as an input; ⋆ **A** makes SA$_2$ query for $\prod_S^j$.
Testpw*(·)	< S,j,C,pw >	For some $l \in \{2,3\}$, • Testpw(S, j, C, pw, l) event occurs.
Testexecpw(·)	< C,i,S,j,pw >	Firstly, **A** makes • An execute query which generates m, pk, ct. • An $H_1(pw)$ query. It returns -γ_S. Then, for $l \in \{2,3\}$, **A** makes • An $H_l(C,S,m,\gamma_S,pk,K)$ query.
Correctpw	-	**A** makes a corrupt query after either of these two events occurs. • Testpw!(C, i, S, pw) event occurs for $\prod_C^i$. • Testpw*(S, j, C, pw) event occurs.
Correctpwexec	-	For some $\{C, i, S, pw\}$, • A Testexecpw(C, i, S, j, pw) event occurs.
Doublepwserver	-	For some $\{S, j, C, pw \neq pw'\}$, The following events occur before any corrupt query. • Testpw*(S, j, C, pw). • Testpw*(S, j, C, pw').
Pairedpwguess	-	For some $\{C, i, S, j\}$, • A Testpw(C, i, S, j, pw) occurs.
- There is no any special input for associated event.		

Algorithm 1 S2 Queries and Answers

- In an execute(C,i,S,j) query, $m = As + e$ where $s \xleftarrow{r} b_\eta^k$ and $e_i \xleftarrow{r} b_\eta$, $pk \xleftarrow{r} D_{pk}$, $ct \xleftarrow{r} D_{ct}$, $K, K''' \xleftarrow{r} \{0,1\}^k$, and $ssk_2^j = ssk_1^i \xleftarrow{r} \{0,1\}^k$.
- In a CA$_0$ query to $\prod_C^i$, $m = As + e$ where $s \xleftarrow{r} b_\eta^k$ and $e_i \xleftarrow{r} b_\eta$.
- In a SA$_1$ query to $\prod_S^j$, $pk \xleftarrow{r} D_{pk}$, $ct \xleftarrow{r} D_{ct}$, $K \xleftarrow{r} \{0,1\}^k$, and $K', ssk_2^j \xleftarrow{r} \{0,1\}^k$.
- In a CA$_1$ query to $\prod_C^i$:
  - As a result of this query, if an Testpw!(C,i,S,pw_C) event occurs, then K''' and ssk_1^i are set to the associated value of Testpw$(C,i,S,pw_C,2)$ and Testpw$(C,i,S,pw_C,3)$, respectively.
  - Otherwise, if $\prod_C^i$ has a partner $\prod_S^j$, $ssk_2^j = ssk_1^i$. Then, $K''' \xleftarrow{r} \{0,1\}^k$.
  - Otherwise, $\prod_C^i$ aborts.
- As a result of a SA$_2$ query, if one of the following conditions is met, it terminates. If not, $\prod_S^j$ aborts.
  - If an Testpw!(S,j,C,pw_C) event occurs, or
  - If $\prod_S^j$ has a partner $\prod_C^i$.
- As a result of an $H_{l\in\{2,3\}}(C,S,m,\gamma_S,pk,K)$, if one of the following conditions is met, the output is obtained with the associated value of the event. If not, the output is chosen from $\{0,1\}^k$.
  - If a Testpw(S,j,C,pw_C,l) event occurs, or
  - If a Testexecpw(C,i,S,j,pw_C) event occurs.

Proof 2 *In **S2**, since m and pk are new due to **S1**, $H_{l\in\{2,3\}}(\cdot)$ is also new. Therefore, the main condition for distinguishing **S1** and **S2** is that **A** queries $H_l(\cdot)$ for $l \in \{2,3\}$. In Algorithm 1, there are two possible cases.*

- *Since **A** does not make any $H_1(pw_C)$, where $-\gamma_S = H_1(pw_C)$, the maximum number of $H_l(\cdot)$ queries **A** can make is $\frac{O(n_o)}{|R_q^k|}$.*
- ***A** makes send(C,i,K') or send(S,j,K''') queries using the actions CA$_0$, CA$_1$, SA$_1$, and SA$_2$ in Algorithm 1. Neither of these queries is the output of an $H_2(\cdot)$ query that would be a correct password guess. Therefore, the maximum probability that **A** can abort the samples is $\frac{O(n_s)}{2^\kappa}$.*

The Claim 3 is satisfied.

***S3**: Unlike **S2**, when an $H_{l\in\{2,3\}}$ is queried, the consistency is not controlled against the query execute. In other words, the event Textexecpw(C,i,S,j,pw_C) is not checked. So, the scheme responds with a random output rather than maintaining consistency with the query execute. In **S2**, let $\varepsilon_3 = Adv^{CCA}_{Kyber\,KEM}(\mathbf{A}) + Adv^{d\text{-}MLWE}_{R_q^k}(T', n_o)$ where $T' = O(T + (n_o + n_s + n_e)T_{exp})$.*

Claim 4 *For any **A**, $Adv^{S2}_{Kyber.PAKE}(\mathbf{A}) \le Adv^{S3}_{Kyber.PAKE}(\mathbf{A}) + \varepsilon_3$*

Proof 3 *Let $E3$ be the occurrence of the event Correctpwexec in **S3**. If $E3$ happens, **S2** and **S3** are distinguishable. In Table 3, if Correctpwexec occurs, the event Textexecpw(C,i,S,j,pw) occurs with two consequences. Given (A,α,φ,ct),*

- *In the query execute, $m = \alpha + (As_1 + e_1)$ and $pk = \varphi + m + \gamma_S$ is set for $s_1 \xleftarrow{r} \beta_q^k$ and $e_1 \xleftarrow{r} \beta_q$. Then, $ct \xleftarrow{r} D_{ct}$ is chosen.*
- *Then, **A** makes query $H_{l\in\{2,3\}}(\cdot)$, where m and pk were obtained by query execute. With query $H_1(pw_C)$, $-\gamma_S = As_h + e_h \in R_q^k$ where $s_h \xleftarrow{r} \beta_q^k$ and $e_h \xleftarrow{r} \beta_q$ obtained. Under these changes, the simulator*

computes $(ct', K') = Kyber.CCAKEM.Enc(pk)$. Then, the obtained (ct', K') is added to the list of possible values.

*Since the Adv of **A** in Kyber KEM, given in Definition 4, is $Adv^{CCA}_{Kyber\ KEM}(\boldsymbol{A})$ and the probability of d-MLWE being resolved is $Adv^{d\text{-}MLWE}_{R^k_q}(T', n_o)$, the Claim 4 is satisfied.*

***S4**: Unlike **S3**, **S4** halts when a correct password guess is made against a $\prod^j_S$ or $\prod^i_C$ instance before any query corrupt. In other words, the event Correctpw happens. Then, **A** automatically succeeds.*

***Claim 5** For any **A**, $Adv^{S3}_{Kyber.PAKE}(\boldsymbol{A}) \leq Adv^{S4}_{Kyber.PAKE}(\boldsymbol{A})$*

***Proof 4** If the event Correctpw occurs,*

- *In a action CA_1 to $\prod^i_C$, if corrupt is not queried after Testpw!(C, i, S, pw_C), **S4** halts and **A** succeeds.*
- *In a query $H_{l\in\{2,3\}}(\cdot)$, if corrupt is not queried after Testpw*(S, j, C, pw_C), **S4** halts and **A** succeeds.*

*The Claim 5 is satisfied as these changes will only increase the win probability of **A** .*

***S5**: Unlike **S4**, **S5** halts when **A** guesses a password against the partner instances $\prod^j_S$ and $\prod^i_C$. In other words, the event Pairedpwguess happens. Then, **A** fails.*

***Claim 6** For any **A**, $Adv^{S4}_{Kyber.PAKE}(\boldsymbol{A}) \leq Adv^{S5}_{Kyber.PAKE}(\boldsymbol{A}) + 4n_s Adv^{d\text{-}MLWE}_{R^k_q}(T', n_o) + Adv^{CCA}_{Kyber\ KEM}(\boldsymbol{A})$*

***Proof 5** If Pairedpwguess occurs, for some C, i, S, j, a Testpw(C, i, S, j, pw_C) occurs. In this event, there is a partnership between $\prod^i_C$ and $\prod^j_S$. Let $d \xleftarrow{r} \{1, 2, \ldots, n_s\}$ be chosen and (A, α, φ, ct) is given. In **S5**, the following changes, given in Algorithm 2, are simulated by **A**.*

Algorithm 2 S5 Changes

- For the d-th send(C, i', S) query to $\prod^{i'}_C$, $m = \alpha$ is set.
- In a send$(S, j, < C, m, \text{seed} >)$, $pk = \varphi + m + \gamma_S$ is set.
- In a send$(C, i', < pk, ct, K >)$, if there is no partner for $\prod^{i'}_C$, the output is 0 and **S5** halts.
- In a send(S, j, K') query to $\prod^j_S$, let $\prod^j_S$ and $\prod^{i'}_C$ be partner after its send$(S, j, < C, m, \text{seed} >)$. If the instances have no partnership after this query and Correctpw is not tested, $\prod^j_S$ aborts.
- Then, **A** makes $H_{l\in\{2,3\}}(\cdot)$ query, where m and pk were obtained with $\prod^{i'}_C$. With $H_1(pw_C)$ query, $-\gamma_S = As_h + e_h \in R^k_q$ is obtained where $s_h \xleftarrow{r} b^k_\eta$ and $e_h \xleftarrow{r} b_\eta$. Under this changes, the simulator computes $(ct', K') = \text{Kyber.CCAKEM.Enc}(pk)$. Then, the obtained (ct', K') is added to the list of possible values.

Since the security of Kyber KEM, given in Definition 4, is $Adv^{CCA}_{Kyber\ KEM}(\boldsymbol{A})$ and the probability of d-MLWE being resolved is $4n_s Adv^{d\text{-}MLWE}_{R^d_q}(\boldsymbol{A})$, the Claim 6 is satisfied.

***S6**: Unlike **S5**, in **S6**, there is an internal password oracle that can know all passwords for a given client/server pair and test the correctness of the provided password.*

***Claim 7** For any **A**, $Adv^{S5}_{Kyber.PAKE}(\boldsymbol{A}) = Adv^{S6}_{Kyber.PAKE}(\boldsymbol{A})$*

***Proof 6** Using the password oracle, **(i.)** All passwords are generated during initialization and special passwords can be tested in the following way. If $pw = pw_C$, the output of testpw(C, pw) is True. Otherwise, the output is False. **(ii.)** All corrupt(U) is accepted and answered. In **S6**, Testpw(C, i, S, pw) for $\prod^i_C$, Testpw(S, j, C, pw) for $\prod^j_S$, and Testpw(C, pw) for password oracle queries are checked whether Correctpw occurs. So, **S5** and **S6** can be completely indistinguishable. The Claim 7 is satisfied.*

*In **S6**, **A** has two ways to gain a non-negligible advantage.*

- *Online dictionary attack: CDF-Zipf model, given in Definition 8, limits the probability of Correctpw event in the proposed Kyber.PAKE since Correctpw event is $\boldsymbol{A}$'s successful obtaining of the password through online dictionary attacks. In other words, $Pr[Correctpw] = C' \cdot n_{op}^{f} + negl(\kappa)$.*
- *A test query: Let $\prod_U^i$ be a fresh instance. Then, $\boldsymbol{A}$ makes a query test(U,i) to $\prod_U^i$. Since the view of $\boldsymbol{A}$ is completely independent of ssk_U^i, $Pr[Suc^{S6}_{Kyber.PAKE}(\boldsymbol{A})|\neg Correctpw] = 1/2$.*

By considering two options, Equation (6) *is obtained.*

$$\begin{aligned} Pr[Suc^{S6}_{Kyber.PAKE}(\boldsymbol{A})] &\leq \overbrace{Pr[Correctpw]}^{C'\cdot n_{op}^{f}} + \overbrace{Pr[Suc^{S6}_{Kyber.PAKE}(\boldsymbol{A})|\neg Correctpw]}^{1/2}\overbrace{Pr[\neg Correctpw]}^{1-C'\cdot n_{op}^{f}} \\ &\leq 1/2(1 + C' \cdot n_{op}^{f}) \end{aligned} \tag{6}$$

According to Equation (2)*, $Adv^{S6}_{AKE}(\boldsymbol{A}) = 2Pr[Suc^{S6}_{Kyber.PAKE}(\boldsymbol{A})] - 1 \leq C' \cdot n_{op}^{f}$. If Equation* (2) *is rewritten by considering Claims* (2)-(7)*, Equation* (7) *is obtained.*

$$\begin{aligned} Adv^{\boldsymbol{S}}_{Kyber.PAKE}(\boldsymbol{A}) &\leq 2|Pr[Suc^{S0}_{Kyber.PAKE}] - \frac{1}{2}| = 2|Pr[Adv^{S0}_{Kyber.PAKE}] - Pr[Adv^{S6}_{Kyber.PAKE}]| \\ &= 2\Big(\overbrace{|Pr[Adv^{S0}_{Kyber.PAKE}] - Pr[Adv^{S1}_{Kyber.PAKE}]|}^{\leq \frac{(n_e+n_s)(n_e+n_s+n_o)}{q^{nk}}} + \overbrace{|Pr[Adv^{S1}_{Kyber.PAKE}] - Pr[Adv^{S2}_{Kyber.PAKE}]|}^{\leq \frac{n_o}{q^{nk}} + \frac{n_s}{2^{\kappa}}} \\ &+ \overbrace{|Pr[Adv^{S2}_{Kyber.PAKE}] - Pr[Adv^{S3=S4}_{Kyber.PAKE}]|}^{Adv^{CCA}_{Kyber\ KEM}(\boldsymbol{A}) + Adv^{d\text{-}MLWE}_{R_q^k}(\boldsymbol{A})} + \overbrace{|Pr[Adv^{S4}_{Kyber.PAKE}] - Pr[Adv^{S5}_{Kyber.PAKE}]|}^{4n_s Adv^{d\text{-}MLWE}_{R_q^k}(\boldsymbol{A}) + Adv^{CCA}_{Kyber\ KEM}(\boldsymbol{A})} \\ &+ \overbrace{|Pr[Adv^{S5}_{Kyber.PAKE}] - Pr[Adv^{S6}_{Kyber.PAKE}]|}^{1/2(1+C'\cdot n_{op}^{f})} \Big) \end{aligned} \tag{7}$$

Since $Adv^{\boldsymbol{S}}_{Kyber.PAKE}(\boldsymbol{A}) \leq C' \cdot n_{op}^{f} + O\Big(\frac{(n_e+n_s)(n_e+n_s+n_o)+n_o}{q^{nk}} + \frac{n_s}{2^{\kappa}} + Adv^{CCA}_{Kyber\ KEM}(\boldsymbol{A}) + n_s Adv^{d\text{-}MLWE}_{R_q^k}(\boldsymbol{A})\Big)$, Theorem 3 is satisfied.

5 IMPLEMENTATION DETAILS AND PERFORMANCE RESULTS

In this section, the implementation results of Kyber.PAKE are presented in terms of cost, CPU cycle, running time, and memory usage.

The implementation of Kyber.PAKE is written in C based on Kyber KEM's reference C codes and PAK design components. The code is available at `https://github.com/afDursun/Kyber-PAKE-C`. A computer with a 2.5 GHz dual-core Intel Core i5 processor and 8 GB RAM is used to obtain performance results. Since the security depends on the same hard problem, MLWE.PAKE scheme (Ren et al., 2023) is chosen for performance evaluation and comparison. Both schemes' parameter sets are recalled in Table 4.

Table 4. Parameter Set

Scheme	Security Level	k	n	q	η	η_1	η_2	(d_u, d_v)	δ
MLWE.PAKE (Ren et al., 2023)	116	2	256	7681	13	x	x	x	$2^{-53.4}$
	177	3	256	7681	8	x	x	x	$2^{-97.4}$
	239	4	256	7681	6	x	x	x	$2^{-131.6}$
Proposed Kyber.PAKE	128	2	256	3329	x	3	2	(10,4)	2^{-131}
	192	3	256	3329	x	2	2	(10,4)	2^{-164}
	256	4	256	3329	x	2	2	(11,5)	2^{-174}

To obtain comparisons in terms of running time, MLWE.PAKE and our implementation are run 1000 times. The median and average CPU cycles of the essential functions/processes for the same security level are given in Table 5. The proposed Kyber.PAKE scheme needs less average and media CPU cycles due to the small size of the parameter set.

Table 5. CPU Cycle Comparision for 128-bit Security Level

	MLWE.PAKE (Ren et al., 2023)		Kyber.PAKE	
Functions/Processes	*Avg.*	*Med.*	*Avg.*	*Med.*
GenMatrix()	31 108	27 997	**24 188**	**22 109**
PolyGetNoise()	4 412	4 112	**3 943**	**3 512**
PolyNtt()	13 429	12 664	**7 798**	**7 443**
PolyvecNtt()	33 170	27 061	**15 024**	**14 121**
PolyvecInvntt()	30 621	26 460	**21 248**	**19 906**
OkcnCon()	17 699	16 058	x	x
OkcnRec()	3 489	3 297	x	x
Kyber.CCAKEM.Enc()	x	x	182 018	165 958
Kyber.CCAKEM.Dec()	x	x	193 497	173 239
C_0	195 201	173 157	**143 497**	**124 864**
S_0	307 547	265 276	**224 537**	**183 024**
C_1	133 436	117 676	256 217	228 652
S_1	40 446	30 603	59 907	57 807

The average running times are given in Table 6, which is constructed by considering common components, scheme phases, hash functions, and reconciliation structures. Due to its parameter set, Kyber.PAKE provides better results in generating pk A with GenMatrix() and hash functions. Since KEM structures such as encapsulation and decapsulation, which have additional components for security, are used in KyberPAKE, it requires more runtime than MLWE.PAKE in terms of reconciliation. Considering the total times on the client and server sides, MLWE.PAKE is better on the client side. One of the reasons is that in MLWE.PAKE, key generation takes place on both the client and server sides, while it is only made on the client side Kyber.PAKE. Different design approaches, reconciliation functions, and parameter sets also affect.

Table 6. Running Times in Microseconds

Scheme	**(Ren et al., 2023)**	**Kyber.PAKE**	**(Ren et al., 2023)**	**Kyber.PAKE**	**(Ren et al., 2023)**	**Kyber.PAKE**
Security Level	**116**	**128**	**177**	**192**	**239**	**256**
GenMatrix()	13.893	9.256	27.504	21.648	49.979	38.713
OkcnCon()	7.058	x	5.920	x	5.293	x
OkcnRec()	1.425	x	1.622	x	1.655	x
Kyber.CCAKEM.Enc()	x	69.133	x	110.894	x	152.360
Kyber.CCAKEM.Dec()	x	72.362	x	117.631	x	177.787
shake128	2.656	2.390	2.422	2.923	3.036	2.397
shake256	13.386	11.328	16.680	16.235	22.904	21.586
C_0	87.456	52.449	112.925	88.894	155.515	141.205
S_0	126.205	71.135	155.530	114.015	202.895	165.042
C_1	50.409	93.443	70.565	150.637	90.342	217.362
S_1	12.942	21.781	16.689	32.918	21.930	42.184
Total Client	138.865	145.892	183.490	239.531	245.857	358.567
Total Server	139.147	**92.916**	172.219	**146.993**	224.825	**207.256**

Comparison results of the two-party schemes which use the lattice-based one-phase PAK approach are also examined in Table 7. Since the hard lattice problems of the schemes are different, only the message sizes that are used in the flows are evaluated while creating Table 7. In Kyber.PAKE, seed, cid, m_{bytes}, and K''' are sent from the client to the server, while pk, ct, K are sent from the server to the client. $seed$, cid, K and K''' are fixed 32-byte. $m_{\text{bytes}}, pk_{\text{bytes}} = k \times 384$ change according to different security levels. For example, for 128-bit security, the size of message transferred from the client to the server is $\text{seed} + \text{cid} + m_{\text{bytes}} + K''' = 32 + 32 + (2 \times 384) + 32 = 864$ bytes where $k = 2$. Similarly, $pk_{\text{bytes}} + ct_{\text{bytes}} + K = (2 \times 384) + 768 + 32 = 1568$ where $k = 2$ is computed from the server to the client.

Remark 2 *The comparisons in Table 5 and Table 6 are conducted by assuming that (Ren et al., 2023) presents approximately the same security levels. Note that Kyber.PAKE will provide better results when the parameters are changed to achieve the same security levels.*

Table 7. A Server and Client Cost Comparison of Lattice-Based PAK PAKE Schemes

Scheme	Hard Problem	Security Level	Client	Server	*Total*
(Ding et al., 2017)	RLWE	76	4136	4256	*8392*
(Gao et al., 2017)	RLWE	82	3904	4000	*7904*
(Yang et al., 2019)	RLWE	206	1864	2592	*4456*
(Ren et al., 2023)	MLWE	116	928	1056	*1984*
		177	1344	1472	*2816*
		239	1760	1888	*3648*
Kyber.PAKE	MLWE	128	**864**	1568	*2432*
		192	**1248**	2272	*3520*
		256	**1632**	3136	*4768*

In bytes

5.1 Kyber.PAKE for Mobile Devices

Using the Kyber.PAKE C codes[*], Java codes[†] are also written to demonstrate the usability of the proposed scheme on mobile devices (Dursun, 2023b). In the implementation, a computer with a 2.5 GHz dual-core Intel Core i5 processor and 8 GB RAM is used as the server. Samsun Galaxy A51 (8 Cores) with 4x 2.3 GHz ARM Cortex-A73 main processor and 4x 1.7 GHz ARM Cortex-A53 co-processor with 2.3 GHz CPU frequency device is utilized as the client. The Kyber.PAKE mobile results in terms of runtime, memory, and CPU usage are given in Table 8, which is obtained by running all the phases of the client and server 1000 times.

Table 8. Implementation Results of Kyber.PAKE on Mobile Device

Security Level	**Phase**	**Running Time***	**Memory Usage**	**CPU Usage**
128	C_0	745.918	104.2 KB	%8
	S_0	880.761	88.6 KB	%10
	C_1	997.569	168.3 KB	%10
	S_1	446.311	0.4 KB	%7
	Total Client	1743.487	272.5 KB	%18
	Total Server	1327.072	89 KB	%17
192	C_0	918.225	148.2 KB	%10
	S_0	945.361	133.7 KB	%11
	C_1	1215.136	211.4 KB	%12
	S_1	611.217	0.4 KB	%8
	Total Client	2133.361	359.6 KB	%22
	Total Server	1556.578	134.1KB	%19
256	C_0	1211.843	177.8 KB	%11
	S_0	1388.745	171.1 KB	%13
	C_1	1811.257	297.2 KB	%14
	S_1	874.413	0.5 KB	%10
	Total Client	3023.1	475 KB	%25
	Total Server	2236.158	171.6 KB	%23

*In microseconds

[*]https://github.com/afDursun/Kyber-PAKE-C
[†]https://github.com/afDursun/Kyber-PAKE-Mobile

6 CONCLUSION

In this paper, we propose a new two-party PAKE version of Kyber KEM. The proposed Kyber.PAKE is constructed as a solution for adapting the algorithms determined as a standard for different purposes and usage areas. The PAKE model is obtained by adapting the traditional PAK design idea, which provides explicit authentication and PFS, to the MLWE problem. By combining the MLWE-based PAK design components and Kyber KEM functions, a password-authenticated shared key is obtained between the parties. The detailed security analysis against dictionary attacks is done using ROM in the BPR model that is constructed by adding the CDF-Zipf model. The proposed PAKE provides two different authentication thanks to the PAKE and KEM combination structure. The run-time, memory, and CPU usage indicate that the Kyber.PAKE scheme can be one of the best choices in post-quantum era security. According to the Java implementation results, the proposed PAKE can also be preferred in the post-quantum security of mobile devices. As a future work, by following the quantum random oracle model (QROM) assumption of Kyber, the security analysis of Kyber.PAKE will be analyzed in the QROM.

ACKNOWLEDGMENT

This research was partially funded by TUBITAK under grant no.121r006.

REFERENCES

Akleylek, S. and Seyhan, K. (2022). Module learning with rounding based key agreement scheme with modified reconciliation. Computer Standards & Interfaces, 79:103549.

Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., and Stehlé, D. (2019). Crystals-kyber algorithm specifications and supporting documentation. NIST PQC Round, 2(4):1–43.

Bellare, M., Pointcheval, D., and Rogaway, P. (2000). Authenticated key exchange secure against dictionary attacks. In International conference on the theory and applications of cryptographic techniques, pages 139–155. Springer.

Bellovin, S. M. and Merritt, M. (1992). Encrypted key exchange: Password-based protocols secure against dictionary attacks.

Bellovin, S. M. and Merritt, M. (1993). Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In Proceedings of the 1st ACM Conference on Computer and Communications Security, pages 244–250.

Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., and Stehlé, D. (2018). Crystals-kyber: a cca-secure module-lattice-based kem. In 2018 IEEE European Symposium on Security and Privacy (EuroS&P), pages 353–367. IEEE.

Dabra, V., Bala, A., and Kumari, S. (2020). Lba-pake: Lattice-based anonymous password authenticated key exchange for mobile devices. IEEE Systems Journal, 15(4):5067–5077.

Ding, J., Alsayigh, S., Lancrenon, J., et al. (2017). Provably secure password authenticated key exchange based on rlwe for the post-quantum world. In Cryptographers' Track at the RSA conference, pages 183–204. Springer.

Ding, R., Cheng, C., and Qin, Y. (2022). Further analysis and improvements of a lattice-based anonymous pake scheme. IEEE Systems Journal, 16(3):5035–5043.

Dursun, A. F. (2023a). Kyber.pake implementation-c codes. `https://github.com/afDursun/Kyber-PAKE-C`. Accessed: 2023-10-25.

Dursun, A. F. (2023b). Kyber.pake implementation-java codes. `https://github.com/afDursun/Kyber-PAKE-Mobile`. Accessed: 2023-10-25.

Gao, X., Ding, J., Li, L., Saraswathy, R., and Liu, J. (2017). Efficient implementation of password-based authenticated key exchange from rlwe and post-quantum tls. Cryptology ePrint Archive.

Hao, F. (2021). Prudent practices in security standardization. IEEE Communications Standards Magazine, 5(3):40–47.

Hao, F. and Ryan, P. Y. (2011). Password authenticated key exchange by juggling. In Security Protocols XVI: 16th International Workshop, Cambridge, UK, April 16-18, 2008. Revised Selected Papers 16, pages 159–171. Springer.

Hao, F. and van Oorschot, P. C. (2022). Sok: Password-authenticated key exchange–theory, practice, standardization and real-world lessons. In Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security, pages 697–711.

Islam, S. H. and Basu, S. (2021). Pb-3paka: Password-based three-party authenticated key agreement protocol for mobile devices in post-quantum environments. Journal of Information Security and Applications, 63:103026.

Jablon, D. P. (1996). Strong password-only authenticated key exchange. ACM SIGCOMM Computer Communication Review, 26(5):5–26.

Liu, C., Zheng, Z., Jia, K., and You, Q. (2019). Provably secure three-party password-based authenticated key exchange from rlwe. In Information Security Practice and Experience: 15th International Conference, ISPEC 2019, Kuala Lumpur, Malaysia, November 26–28, 2019, Proceedings 15, pages 56–72. Springer.

MacKenzie, P. (2002). The pak suite: Protocols for password-authenticated key exchange. Contributions to IEEE P, 1363(2).

NIST (2022a). Post-quantum cryptography. `https://csrc.nist.gov/Projects/post-quantum-cryptography`. Accessed: 2023-09-25.

NIST (2022b). Post-quantum cryptography- selected algorithms 2022. `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`. Accessed: 2023-09-25.

Ott, D., Peikert, C., et al. (2019). Identifying research challenges in post quantum cryptography migration and cryptographic agility. arXiv preprint arXiv:1909.07353.

Peikert, C. et al. (2016). A decade of lattice cryptography. Foundations and trends® in theoretical computer science, 10(4):283–424.

Ren, P., Gu, X., and Wang, Z. (2023). Efficient module learning with errors-based post-quantum password-authenticated key exchange. IET Information Security, 17(1):3–17.

Seyhan, K. and Akleylek, S. (2023). A new password-authenticated module learning with rounding-based key exchange protocol: Saber. pake. The Journal of Supercomputing, pages 1–38.

Shin, S. and Kobara, K. (2012). Efficient augmented password-only authentication and key exchange for ikev2. Technical report.

Wang, D., Cheng, H., Wang, P., Huang, X., and Jian, G. (2017a). Zipf's law in passwords. IEEE Transactions on Information Forensics and Security, 12(11):2776–2791.

Wang, D., Cheng, H., Wang, P., Huang, X., and Jian, G. (2017b). Zipf's law in passwords. IEEE Transactions on Information Forensics and Security, 12(11):2776–2791.

Wu, T. (1998). The secure remote password protocol. In NDSS, volume 98, pages 97–111. Citeseer.

Yang, Y., Gu, X., Wang, B., and Xu, T. (2019). Efficient password-authenticated key exchange from rlwe based on asymmetric key consensus. In International Conference on Information Security and Cryptology, pages 31–49. Springer.