Accepted by IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, 2026,DOI: 10.1109/TDSC.2026.3753541

qaPAKE: Quantum-Safe Anonymous Password-Only Authenticated Key Exchange for Mobile Devices

Jingwei Jiang and Ding Wang

***Abstract*—Secure communication and privacy-preserving user authentication are essential for modern resource-constrained mobile devices. Anonymous password-authenticated key exchange (aPAKE) enables two parties to mutually authenticate and establish a high-entropy session key using a low-entropy password while preserving anonymity. Existing aPAKE protocols struggle to balance quantum resistance and password-only authentication. Lattice is promising for building quantum-safe aPAKE, yet constructing a password-only aPAKE over lattices from a classical aPAKE remains challenging. Most aPAKE protocols require users to maintain additional identifiers beyond a username and password or rely on auxiliary devices, while a series of critical components are inherently incompatible with aPAKE protocols.**

In this paper, we address this challenge by providing a *brand new technical route* to build a secure and efficient quantum-resistant password-only aPAKE. Specifically, we first propose a lightweight verifiable partially oblivious pseudorandom function (VPOPRF) over lattices. Our VPOPRF preserves password privacy while locating credentials *without* indexes. Then, we construct the first lattice-based password-only aPAKE, named qaPAKE. Our qaPAKE enables anonymous authentication only with a username and password (i.e., *without* any device restrictions). In particular, we formally analyze its security and mutual anonymity (i.e., both user and server anonymity) in the classical random oracle model and assess its quantum resistance. Our qaPAKE runs in 139.04 ms (honest-but-curious) and 573.71 ms (malicious) with 100 registered credentials on a common consumer smartphone, and well balances the demanding requirements of quantum-resistance and password-only properties.

***Index Terms*—Quantum-safe, Lattice, Password authentication, Anonymous, Mobile Devices.**

I. Introduction

SINCE the advent of computers, authentication by entering a username and password has been widely used. Billions of password authentications take place daily in cyberspace, and they are unlikely to be replaced in the near future [1], [2]. Passwords are easy to understand and deploy, and allow users to authenticate using only a username and password without relying on auxiliary authentication devices. Password-authenticated key exchange (PAKE) [3], [4] allows two parties, such as a user and a server [5], [6], [7], [8], to establish a high-entropy session key from a low-entropy password.

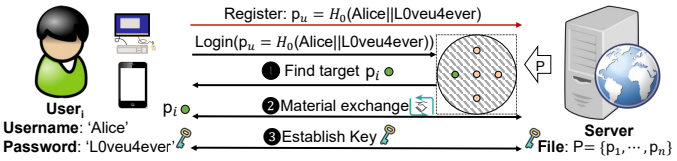


Fig. 1: An exemplary overview of a password-only anonymous PAKE. For better illustration, the red arrow denotes a secure channel, and the **black** arrow denotes a public channel. The user U computes a credential cre_u, and the server S masks the credential set $\mathcal{T}$. ❶ S sends $\mathcal{T}$ to U, and U retrieves the credential associated with cre_u. ❷ U and S perform mutual authentication and exchange key material. ❸ U and S derive the same session key and establish a secure channel.

Anonymous PAKE (aPAKE) extends PAKE by additionally protecting the privacy of users and server-side authentication files. In the mutual-anonymity notion [9] of aPAKE, the server cannot associate an authentication session with a user's corresponding authentication file, while the user learns no information about the server's authentication-file set beyond whether authentication succeeds. aPAKE serves as a fundamental primitive for applications, such as the Internet of Things [10] and Wi-Fi hotspot authentication [11], and is currently being standardized by ISO/IEC [12].

Existing aPAKE protocols can be broadly categorized into two types. The first type uses a credential cre_u as the authentication material [13], [14], [15]. The user U protects cre_u with a password pw_u and retrieves it during authentication. However, such aPAKEs require U to maintain cre_u as additional persistent client-side state. While benefiting from the security and performance advantages of anonymous credential systems [16], this comes at the cost of password convenience.

The second type uses passwords directly as authentication material, and are referred to as password-only aPAKEs [17], [18], [19], [20], [21], [9]. In this work, password-only means that the user maintains no persistent authentication state beyond the username un_u and password pw_u. In this setting, U holds un_u and pw_u, and the server S maintains a credential set $\mathcal{T}$. The two parties anonymously verify that the user's password-derived authentication material matches an entry in $\mathcal{T}$ and then establish a session key for a secure channel.

Viet et al. [17] proposed the first password-only aPAKE based on oblivious transfer (OT) [29]. Subsequent password-only aPAKEs [18], [19], [20], [21] followed the OT-based design approach. However, these protocols require U to maintain an additional index inx_u, which identifies the position of the user's authentication file in $\mathcal{T}$. Such an index inx_u constitutes persistent client-side authentication state and therefore does not satisfy the strict password-only requirement. Specifically,

This work was supported in part by the National Cryptologic Science Fund of China (2026NCSF01022); and in part by the National Natural Science Foundation of China under Grant (62502241); and in part by the Tianjin Natural Science Foundation Project under Grant (25JCJQJC00230). (Corresponding author: Ding Wang.)

Jingwei Jiang and Ding Wang are with the College of Cryptology and Cyber Science, Nankai University, Tianjin 300350, China, and also with the State Key Laboratory of Cryptology, Beijing 100878, China, also with the Tianjin Key Laboratory of Network and Data Security Technology, Nankai University, Tianjin 300350, China, and also with the Beijing Key Laboratory of Post-Quantum Cryptography, Beijing 100083, China. (Email: jiangjingwei@nankai.edu.cn; wangding@nankai.edu.cn).

TABLE I: Comparison of various aPAKE protocols.†

Protocol	Year	Storage Architecture	User Cognition	Assumption	Model	Tool	Type	R/F	Quantum Security	User		Server	
										Comp.	Comm.	Comp.	Comm.
Shahidinejad et al. [22]	2025	Device	un+pw	RO	Game	Kyber PKE	CS	2/3	Y	○	○	○	○
Mani et al. [23]	2025	Device	un+pw	RO	Game	RLWE KE	CS	1/2	Y	○	○	○	○
Ding et al. [24]	2022	Device	un+pw	RO	Game	RLWE KE	CS	2/3	Y	○	○	○	○
Dabra et al. [25]	2020	Device	un+pw	RO	Game	RLWE KE	CS	2/3	Y	○	○	○	○
Feng et al. [26]	2018	Device	un+pw	RO	Game	RLWE KE	CS	2/3	Y	○	○	○	○
Our qaPAKE	-	none	un+pw	RO	Game	OPRF+KE	PO	2/4	Y	○	○	○	●
Liu et al. [27]	2026	none	un+pw	RO	Game	OPRF+Sign+DH	PO	2/3	N	○	○	○	○
Vasco et al. [9]	2019	none	pw	RO	Game	OPRF	PO	3/6	N	○	○	○	●
	2019	none	pw	RO	Game	IBE	PO	3/6	N	○	○	●	●
Hu et al. [20]	2017	none	un+pw	Std	UC	SPHF+Commit	PO	2/3	N	○	○	●	●
Hu et al. [21]	2017	none	un+pw	Std	UC	SPHF	PO	2/3	N	○	○	●	●
Zhang et al. [13]	2016	Device	pw	RO	Game	MAC	PP	2/3	N	○	○	○	○
Yang et al. [15]	2010	Device	pw	RO	Game	BBS Sign	PP	2/3	N	○	○	○	○
Yang et al. [14]	2009	Device	pw	RO	Game	CL Sign	PP	2/3	N	○	○	○	○
Shin et al. [19]	2009	none	un+pw	RO	Game	DH-like KE	PO	2/3	N	●	○	●	●
Yang and Zhang [18]	2008	none	un+pw+G	RO	Game	OT+KE	PO	2/4	N	●	○	●	●
Shin et al. [28]	2007	none	un+pw+G	RO	Game	OT	PO	1/2	N	●	●	●	●
Viet et al. [17]	2005	none	un+pw	RO	Game	OT	PO	2/3	N	●	○	●	●

† un=username; pw=password; G=group; RO=random oracle; Std=standard; Sign=signature; OT=oblivious transfer; MAC=message authentication code; SPHF=smooth projective hash function; KE=key exchange; OPRF=oblivious pseudorandom functions; IBE=identity-based encryption; PP=password protected; PO=password-only; CS=client stores; R/F=round/flow; Y=yes; N=no. Comp.=computation; Comm.=communication. The "Device" indicates the need for additional devices to store secrets. The computation and communication demonstrate whether the online phase cost is linearly correlated ($\mathcal{O}(n)$) with the credential database, where n is the size of the credential database. ● denotes linearly correlated; ○ denotes independent. ○ is preferable over ●.

maintaining a fixed inx creates additional overhead for managing $\mathcal{T}$ [9]. For instance, when U leaves, S must update inx_u of all remaining users after revoking pw_u. Moreover, the inx_u update log inadvertently records U's revocation behavior.

Vasco et al. [9] proposed a password-only aPAKE employing oblivious pseudorandom functions (OPRF), eliminating the need for U to maintain inx_u. Still, this protocol requires S to store users' passwords in plaintext, exposing them directly if the server is compromised. Moreover, because multiple users may choose the same password, the server cannot uniquely associate a plaintext password with a particular account, which complicates account-specific password revocation and updates.

Moreover, the rapid advancements in quantum computing [30] pose a significant threat to password-only aPAKE [9], as many rely on classical assumptions, such as large integer factorization and discrete logarithms, which can be efficiently solved by Shor's algorithm [31]. Standards organizations (e.g., IETF and NIST) are actively preparing the transition to post-quantum cryptography [32], [33]. Recently, NIST-IR-8547-ipd [34] outlines a transition timeline for deprecating and eventually disallowing quantum-vulnerable cryptographic primitives by 2035. Lattice-based cryptography has become a leading approach to general-purpose post-quantum cryptography, as reflected in NIST's standardization process [35].

Dozens of quantum-safe aPAKEs [25], [24] and password-based protocols [36], [37], [38] have been proposed over lattices. However, existing lattice-based aPAKEs require users to maintain auxiliary client-side state (e.g., credentials, randomness, or salts), and therefore do not satisfy the password-only requirement in this work. To the best of our knowledge, no existing aPAKE simultaneously provides quantum-resistance, mutual anonymity, and password-only authentication without persistent auxiliary client-side state. This raises the question:

Is it possible to construct a lattice-based aPAKE that simultaneously achieves quantum resistance, mutual anonymity, and password-only authentication?

Motivations. Existing lattice-based aPAKEs [26], [25], [24] require users to maintain protocol-specific persistent authentication state. Consequently, authenticating from a new or replacement device requires this state to be transferred, synchronized, or recovered. This requirement weakens the cross-device convenience of password authentication and introduces additional state-management and recovery mechanisms.

Moreover, conventional pseudonym-based database lookup does not directly satisfy the user-anonymity requirement in our setting. Although such lookup enables efficient retrieval without exposing the user's explicit identity, it still allows the server to associate an authentication attempt with a specific user record. Such explicit linkage is undesirable when mutual anonymity is required. Our motivation is therefore to construct an aPAKE that simultaneously preserves mutual anonymity and post-quantum security, while allowing users to authenticate with only a username and password and without maintaining additional persistent client-side authentication state.

Challenges. Although Vasco et al. [9] proposed a password-only aPAKE based on classical hardness problems, designing a quantum-safe password-only aPAKE protocol remains a significant challenge. The two primary reasons are as follows.

Firstly, the OPRF-based password-only aPAKE protocol [9] requires the server S to store registered users' plaintext passwords to enable offline password blinding. Still, plaintext password databases are attractive targets for external attackers and malicious insiders [55]. Once plaintext passwords are exposed, adversaries can impersonate users without executing offline guessing attacks. Incidents involving the storage of vast numbers of plaintext passwords and password breaches further highlight the risks associated with such leaks [56], [57].

In addition, Wang et al. [58] point out that the ten most popular passwords in each evaluated dataset account for $0.79\% \sim 10.44\%$ of its password entries. Thus, the server may store repeated password values in its database. The compromise of one victim's password can facilitate guessing attempts against other users with the same password. The challenge

TABLE II: Comparison among recent OPRF constructions.†

Scheme	Round	Hardness	PRF	Building block	Verify	QS	Model			
							User	Server	Assumption	Security
Esgin et al. (CRYPTO'26) [39]	1	MLWE	BP14	Hash+HE+ZK	✓	✓	●	●	ROM	Game
Alamati et al. (CRYPTO'24) [40]	1	AM	NR	OT+VOLE+SS	×	✓	○	○	STA	SC
Heimberger et al. (AsiaCCS'24) [41]	2	CSIDH	NR	OT	×	✓	○	○	ROM	Game
Albrecht et al. (EUROCRYPT'24) [42]	1	MLWE+MSIS	BIP18	FHE+ZK	✓	✓	●	●	ROM	Game
Dinur et al. (CRYPTO'21) [43]	1	LPN	BIP18	OT+MPC	×	✓	○	○	STA	SC
Albrecht et al. (PKC'21) [44]	1	RLWE+SIS	BP14	HE+ZK	✓	✓	●	●	CRS	SC
Boneh et al. (ASIACRYPT'20) [45]	2	CSIDH	NR	OT+ZK	✓	✓	●	●	ROM	UC
Our qaPAKE	1	RLWE	Hash	Hash+HE+ZK	✓	✓	●	●	ROM	Game
Tyagi et al. (EUROCRYPT'22) [46]	1	CSIDH	DY	3HashDH+ZK	✓	×	●	●	ROM	Game
Hazay (J.Cryptol'18) [47]	2	DDH	NR	HE+ZK	✓	×	●	●	STA	SC
Jarecki et al. (EUROCRYPT'18) [3]	1	CDH	HashDH	2HashDH+Auth	✓	×	●	●	ROM	UC
Camenisch and Lehmann (EuroS&P'17) [48]	1	DCR+DHI	DY	HE+ZK	✓	×	●	●	CRS	UC
Jarecki et al. (ACNS'17) [49]	1	CDH	HashDH	2HashDH+SS+Auth	✓	×	●	●	ROM	UC
Everspaugh et al. (UsenixSec'15) [50]	1	BDDH	HashDH	BP+ZK	✓	×	●	●	CRS	Game
Jarecki et al. (ASIACRYPT'14) [51]	1	DCR+RSA	HashDH	HE+SS+ZK	✓	×	●	●	CRS	UC
Jarecki and Liu (TCC'09) [52]	2	DHI	DY	HE+ZK	✓	×	●	○	STA	SC
Hazay and Lindell (TCC'08) [53]	1	DCR	NR	OT	×	×	●	●	STA	SC
Freedman et al. (TCC'05) [54]	1	DDH	NR	OT	×	×	●	○	STA	SC

† DH=Diffie-Hellman; QS=Quantum security; AM=Alternating-moduli; NR=Naor-Reingold model; OT=Oblivious transfer; VOLE=Vector oblivious linear evaluation; SS=Secret sharing; STA=Standard; SC=Secure computation; CSIDH=Commutative supersingular isogeny DH; ROM=Random oracle model; MLWE=Module Learning with Errors; MSIS=Module Short Integer Solution; FHE=Fully Homomorphic Encryption; ZK=Zero-Knowledge; DY=Dolev-Yao Model; LPN=Learning Parity with Noise; MPC=Multi-Party Computation; RLWE=Ring Learning with Errors; HE=Homomorphic Encryption; CRS=Common Reference String; DDH=Decisional DH; CDH=Computational DH; UC=Universally Composable; DCR=Decisional Composite Residuosity; DHI=DH inversion; Auth=Authentication; BDDH=Bilinear Decisional DH; BP=Bilinear pairing; RSA=Rivest-Shamir-Adleman. ✓=Yes; ×=No; ● =Malicious; ○ =Honest. Our work is suitable for one-time password rerandomization, and its ZK proof is simpler than Albrecht et al. [42].

is therefore to replace plaintext passwords with server-key-dependent credential material that users can reconstruct with server assistance from a username and password, while preserving anonymous authentication and offline preprocessing.

Secondly, the crux of building a password-only aPAKE is enabling U to reconstruct cre_u and locate the corresponding entry in a masked representation of $\mathcal{T}$ using only un_u and pw_u, without maintaining a location index or revealing the selected entry to the server. OPRFs are employed to derive password-dependent cryptographic material in aPAKE [9] and other password-based protocols [51], [3], [59]. However, the lattice-based OPRF of Albrecht et al. [44] uses non-linear bit decomposition in its input encoding, complicating the corresponding zero-knowledge proofs. With the parameters and proof instantiation in [44], a rough lower-bound calculation gives more than 2^{40} bits of communication per repetition.

In addition, lattice-based OPRFs [42], [44] require careful treatment of verifiability when integrated into aPAKE. Here, this entails checking the validity of the blinded request and the consistency of the response with the OPRF key commitment, while keeping the password-derived input and the server's OPRF key private during authentication. Furthermore, the broadcast mechanism requires a separate consistency check to ensure that all ciphertexts within each batch encrypt the same batch-specific verification key pk_s, preventing credential-dependent key tagging by a malicious server. Session-key establishment must use fresh, session-specific secret material rather than a secret shared across registered users.

A. Related work

Anonymous PAKE. Viet et al. [17] proposed the first aPAKE in 2005, with a concrete instantiation based on oblivious transfer (OT) [29]. Subsequently, Yang et al. [18] improved the aPAKE by addressing replay attacks and weaknesses against active adversaries, and proposed an improved OT-based aPAKE with enhanced security. Shin et al. [19] further advanced aPAKE efficiency for practical deployment. Additionally, Hu et al. [21] proposed an aPAKE using smooth projective hashing under the standard model. However, such aPAKEs rely on a heuristic security model and fail to adequately capture the application scenarios relevant to aPAKE.

Vasco et al. [9] proposed a formal security model for aPAKE that captures user and server anonymity and introduced a password-only aPAKE. However, such an approach requires S to store plaintext passwords, a practice shown to be highly vulnerable given the frequent occurrences of large-scale password leaks [56], [57]. Another approach to implementing aPAKE assumes that U stores a persistent cryptographic credential. Yang et al. [14] proposed the first instance of the user-storage model with CL signatures [16]. Subsequently, Yang et al. [15] and Zhang et al. [13] proposed such constructions employing BBS signatures [60] and algebraic MACs [61], respectively.

Driven by the advantages of lattices, a series of quantum-safe aPAKEs [26], [25], [24] have been proposed, building on Ding et al.'s key exchange [62] and refined to mitigate signal leakage attacks [63]. Table I provides a comparative overview of aPAKEs. Related lattice-based authentication research addresses both security and application requirements. For mobile communication, Dharminder and Chandran integrated biometric fuzzy extraction [64], with Kumar et al. [65] addressing its signal leakage weakness. Broader developments cover unilateral and bilateral authentication and lattice-based key exchange [23], [66], while application-specific designs address authentication in drone [67] and satellite systems [68].

These related protocols maintain client-side authentication material, and several additionally require biometric inputs. Their anonymity goals also differ from ours. Taken together, the surveyed protocols [27], [9] leave a gap between password-

only authentication and quantum resistance. Among the prior protocols compared in Table I, password-only aPAKEs do not provide quantum resistance, whereas quantum-resistant constructions require persistent client-side authentication material.

Moreover, Vasco et al.'s OPRF-based construction [9] requires server-side plaintext password storage. Our objective is therefore to construct a quantum-resistant aPAKE that preserves mutual anonymity while requiring users to maintain only a username and password. To this end, qaPAKE combines OPRF-based credential reconstruction, local selection from a masked credential database, and fresh session-specific key establishment, without storing plaintext passwords on the server. This motivates our review of lattice-based OPRFs below.

OPRF. An oblivious pseudorandom function (OPRF) [50], [44] extends the concept of a pseudorandom function (PRF). The OPRF allows the user U to obtain a pseudorandom output $F_K(x)$ without revealing its input x to the server S, while S learns nothing about x, ensuring mutual privacy. Specifically, U provides x, and S, holding a secret key K, computes $F_K(x)$. During this process, x is *blinded*, preventing S from learning x, while S guarantees that U receives the correct $F_K(x)$ without exposing K. This unique capability makes OPRFs indispensable in privacy-sensitive contexts [50], [51], [3], [59]. U can transform a low-entropy password pw into a high-entropy session key sk with K, thereby significantly reducing the threat of offline password guessing attacks.

Given the significance of OPRFs, numerous constructions have been proposed, either based on classical assumptions [54], [52], [50], [46] or on post-quantum assumptions [40], [41], [42], [43], [44], [45]. Table II provides an overview of the current OPRFs. Albrecht et al. [44] proposed the first lattice-based OPRF, establishing a brand route for subsequent OPRF [42], [39]. Informally, U blinds the secret input using randomness, S rerandomizes the secret using its key, and U achieves unblind operation with the server's long-term public key to obtain a pseudorandom output $F_K(x)$. Still, such OPRFs incur prohibitively high overhead for anonymous password-authenticated key exchange (aPAKE) protocols. To the best of our knowledge, no lattice-based OPRF suitable for lightweight password-only aPAKE has yet been proposed.

B. Our contributions

In this work, we propose a quantum-safe anonymous Password-only Authenticated Key Exchange, by constructing a lightweight OPRF over lattices, named qaPAKE. In summary, our contributions are four-fold:

- **Lattice-based OPRF**. We design a lightweight verifiable partially OPRF (VPOPRF) based on lattice assumptions. The proposed VPOPRF allows a user to input only a password-derived secret and interact with the server to obtain a PRF evaluation with minimal computational overhead and a zero knowledge proof. Our construction ensures that an adversary cannot recover the user's secret from the pseudorandom function output, nor can it predict the PRF value corresponding to any secret. With the public input fixed as $\mathcal{I} = \perp$, our VPOPRF provides a lightweight password hardening mechanism that is well suited for password authentication protocols.
- **Quantum-resistant password-only aPAKE**. We provide a framework for lattice-based password-only aPAKE against quantum attacks. Then, we construct an instance of password-only aPAKE based on OPRF-based credential reconstruction, local selection from a masked broadcast, and lattice-based key exchange, named qaPAKE. To the best of our knowledge, our qaPAKE is the *first* quantum-safe password-only aPAKE protocol, allowing users to achieve anonymous login using only conventional human memorable username and password patterns.
- **Security analysis**. We conduct a comprehensive formal security analysis of qaPAKE in the random oracle model under state-of-the-art password distribution models [69] and anonymous authentication models [9]. In addition, we provide a detailed parameter selection analysis and present two concrete parameter sets that achieve quantum security levels of 92 bits and 128 bits, respectively, suitable for lightweight applications and high-security requirements. Our analysis demonstrates that qaPAKE preserves authentication security and anonymity while resisting a wide range of attacks from adversaries equipped with either classical or quantum computing capabilities.
- **Performance comparison**. We analyze the computational and communication costs of password-only aPAKE constructed using four different methods [17], [18], [19], [9]. We evaluate qaPAKE on a Huawei P30 under honest-but-curious and malicious settings. Across databases of 10–1,000 credentials, client online computation remains near 65.6 ms and 401.2 ms, respectively. We compare widely aPAKEs (including password-protected aPAKE [14], [15], [13], aPAKE with storing secrets on the client [26], [25], [24], and password-only aPAKE [17], [18], [19], [20], [21], [9]). Comparison results show that qaPAKE provides convenience and quantum resistance.

II. Preliminaries

Notations. Let $\mathbb{Z}$ and $\mathbb{R}$ denote the set of all integers and the set of real numbers, respectively. For any integer $q \geq 2$, there is a polynomial $R_q := \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ in the power-of-two cyclotomic ring. Let upper-case letter $\mathbf{A}$ and lower-case bold letter $\mathbf{x}$ denote matrices and vectors, respectively. We write $x \leftarrow \mathcal{D}$ to denote the sampling of x according to distribution $\mathcal{D}$. We write $x \leftarrow S$ to indicate sampling uniformly at random from a finite set S. λ is the security parameter.

A. Gaussian Sampling and Lattices

Definition 1 ([70]). *Let $\Lambda_q(\mathrm{A}) = \{\mathrm{A}\boldsymbol{s} \mid \boldsymbol{s} \in \mathbb{Z}_q^n\}$ denote an m-dimensional lattice with the basis $\mathrm{A} \in \mathbb{Z}_q^{m\times n}$ for $m \geq n\log q$, and there is $\det(\Lambda) = \sqrt{\det(\mathrm{B}^T\mathrm{B})}$, where B is a basis matrix.*

Definition 2 (**Gaussian distributions** [71]). *For a standard deviation $\sigma > 0$, define the discrete Gaussian distribution over an integer lattice $\Lambda \subseteq \mathbb{Z}^m$ centred at $\mathbf{c} \in \mathbb{R}^m$ with parameter σ to be: $D_{R,\sigma}(\mathbf{x}) = \rho_{\mathbf{c},\sigma}(\boldsymbol{x})/\rho_{\mathbf{c},\sigma}(\Lambda)$, where $\boldsymbol{x} \in \Lambda$, $\rho_{\mathbf{c},\sigma}(\boldsymbol{x}) = e^{-\pi||\mathbf{x}-\mathbf{c}||^2/\sigma^2}$, and $\rho_{\mathbf{c},\sigma}(\Lambda) = \sum_{\boldsymbol{x}\in\Lambda}\rho_{\mathbf{c},\sigma}(\boldsymbol{x})$.*

Definition 3 (Decision-RLWE$_{n,q,\chi,m}$ [70]). *For a prime integer q, integers $m, n > 0$, and a noise distribution $\mathcal{X}$ over R_q,*

sample $\mathbf{A} \leftarrow R_q^{m\times n}, \boldsymbol{s} \leftarrow R_q^{n\times 1}, \boldsymbol{e} \leftarrow \chi^{m\times 1}, \boldsymbol{b'} \leftarrow R_q^m$. *The Decision-RLWE*$_{n,q,\chi,m}$ *(DRLWE) problem is to distinguish between* $(\mathbf{A}, \mathbf{b}) \in R_q^{m\times n} \times R_q^{m\times 1}$ *and* $(\mathbf{A}, \boldsymbol{b'}) \in R_q^{m\times n} \times R_q^{m\times 1}$, *where* $\mathbf{b} = \mathbf{A} \cdot \boldsymbol{s} + \boldsymbol{e} \mod q$. *For any probabilistic polynomial time (PPT) adversary* $\mathcal{A}$, *the two distinct distributions are computationally indistinguishable, i.e.,* $Adv_{\mathcal{A}}^{\mathsf{DRLWE}}(\lambda) = |\mathsf{Pr}[\mathcal{A}(q, m, n, \mathcal{X}, \mathbf{A}, \mathbf{b})] - \mathsf{Pr}[\mathcal{A}(q, m, n, \mathcal{X}, \mathbf{A}, \mathbf{b'})]| \leq \varepsilon(\lambda)$.

Definition 4 (Rounding [72])**.** *Let* $q, m, n \in \mathbb{Z}^+$. *For* $\mathbf{A} \leftarrow R_q^{m\times n}$, *the rounding algorithm* F *is deterministic for the "rounding" function* $\lfloor\cdot\rceil : R_q \rightarrow R_p$ *that enables to map* $\mathbf{x} \rightarrow \mathbf{u}$ *such that* $F(\mathbf{x}) = \mathbf{A} \cdot \mathbf{u}$.

Definition 5 (LWR [72])**.** *For* $p, q, m, n \in \mathbb{Z}^+$, *the LWR problems state that the two distributions* $(\mathbf{A}, \lfloor \mathbf{A} \cdot \mathbf{s} \rfloor_p)$ *and* $(\mathbf{A}, \lfloor \mathbf{u} \rfloor_p)$ *are computationally indistinguishable, where* $\mathbf{A} \leftarrow R^{m\times n}, \mathbf{s} \leftarrow R^n$ *and* $\mathbf{u} \leftarrow R^m$. *The advantage of* $\mathcal{A}$ *solving LWR* $_{p,q,m,n}$ *is defined as follows* $\mathrm{Adv}^{\mathrm{LWR_{p,q,m,n}}} = |\Pr[\mathcal{A}(\mathbf{A}, \lfloor \mathbf{A} \cdot \mathbf{s} \rfloor_p) \rightarrow 1] - \Pr[\mathcal{A}(\mathbf{A}, \lfloor \mathbf{u} \rfloor_p) \rightarrow 1]|$ *of* $\mathcal{A}$.

B. Lattice-based Encryption

In this section, we review a semantically secure lattice-based encryption scheme [70], comprising a set $(\mathsf{EncKG}, \mathsf{Enc}, \mathsf{Dec})$.

Definition 6 ([70])**.** *Let* λ *be a security parameter.* $seed$ *is a seed of key generation. A lattice-based encryption scheme must contain the following three polynomial algorithms:*

- $(sk, pk) \leftarrow \mathsf{EncKG}(1^\lambda, seed)$ *is a key generation algorithm. On input* $(\lambda, seed)$, *it outputs a key pair* (sk, pk).
- $c \leftarrow \mathsf{Enc}(pk, m)$ *is an encryption algorithm. On input* pk *and message* m, *it outputs ciphertext* c.
- $m \leftarrow \mathsf{Dec}(sk, c)$ *is a decryption algorithm. On input a ciphertext* c *and secret key* sk, *it outputs* m.

C. Lattice-based Key-Encapsulation Mechanism

A key-encapsulation mechanism (KEM) allows two parties to establish a shared secret over a public channel. In this work, we employ the ML-KEM, standardized by NIST in FIPS 203 [73]. ML-KEM consists of three probabilistic polynomial-time algorithms $\mathsf{KEM} = (\mathsf{KEM.Gen}, \mathsf{KEM.Encaps}, \mathsf{KEM.Decaps})$.

Definition 7 (ML-KEM [73])**.** *Let* λ *be the security parameter. A key-encapsulation mechanism consists of the following three polynomial-time algorithms:*

- $(dk, ek) \leftarrow \mathsf{KEM.Gen}(1^\lambda)$ *is the key-generation algorithm. On input the security parameter* λ, *it outputs a decapsulation key* dk *and an encapsulation key* ek.
- $(ct, z) \leftarrow \mathsf{KEM.Encaps}(ek)$ *is the encapsulation algorithm. On input an encapsulation key* ek, *it outputs a ciphertext* ct *and a shared secret* z.
- $z' \leftarrow \mathsf{KEM.Decaps}(dk, ct)$ *is the decapsulation algorithm. On input a decapsulation key* dk *and a ciphertext* ct, *it outputs a shared secret* z'.

For an honestly generated key pair and ciphertext, the correctness of the KEM requires

$$\Pr\left[z' = z \;\middle|\; \begin{array}{l}(dk, ek) \leftarrow \mathsf{KEM.Gen}(1^\lambda),\\ (ct, z) \leftarrow \mathsf{KEM.Encaps}(ek),\\ z' \leftarrow \mathsf{KEM.Decaps}(dk, ct)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

D. Lattice-based Signature

In this section, we recall ML-DSA, a signature scheme standardized by NIST in FIPS 204 [74], as follows.

Definition 8 (ML-DSA [74])**.** *Let* λ *be the security parameter. A digital signature scheme consists of the following three polynomial-time algorithms:*

- $(sk, pk) \leftarrow \mathsf{SIG.Gen}(1^\lambda)$ *is the key-generation algorithm. On input the security parameter* λ, *it outputs a signing key* sk *and a corresponding verification key* pk.
- $\sigma \leftarrow \mathsf{SIG.Sign}(sk, m)$ *is the signing algorithm. On input a signing key* sk *and a message* $m \in \{0,1\}^*$, *it outputs a signature* σ.
- $b \leftarrow \mathsf{SIG.Ver}(pk, m, \sigma)$ *is the verification algorithm. On input* pk, *a message* m, *and* σ, *it outputs* $b = 1$ *if the signature is valid and* $b = 0$ *otherwise.*

For an honestly generated key pair and signature, the following verification failure probability is negligible:

$$\Pr\left[0 \leftarrow \mathsf{SIG.Ver}(pk, m, \sigma) \;\middle|\; \begin{array}{l}(sk, pk) \leftarrow \mathsf{SIG.Gen}(1^\lambda)\\ \sigma \leftarrow \mathsf{SIG.Sign}(sk, m)\end{array}\right].$$

E. Symmetric Encryption

We define the algorithm package of symmetric encryption used in qaPAKE as follows. The encryption key is supplied externally and is derived from the user's credential material.

Definition 9 (Symmetric encryption)**.** *Let* λ *be a security parameter, and let* $\mathcal{K}_\lambda = \{0,1\}^{\kappa(\lambda)}$ *denote the key space, where* $\kappa(\lambda)$ *is the key length. A symmetric encryption scheme consists of two polynomial-time algorithms defined as follows.*

- $c \leftarrow \mathsf{SEnc}(k, m)$ *is a randomized encryption algorithm. On input a secret key* k *and a message* m *in the message space, it outputs a ciphertext* c.
- $m' \leftarrow \mathsf{SDec}(k, c)$ *is a deterministic decryption algorithm. On input a secret key* k *and a ciphertext* c, *it outputs a message* m' *or the failure symbol* $\perp$.

F. Commitment and Zero-Knowledge proof

A commitment scheme allows a user to commit to a vector $\mathbf{x}$ and, later, provide a concise opening. The commitment algorithm guarantees the completeness of zero-knowledge proofs. The definition of a commitment scheme is as follows.

Definition 10. *A commitment scheme consists of a tuple* $COM = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Verify})$ *defined as follows:*

- $\mathsf{ck} \leftarrow \mathsf{Com.Setup}(1^\lambda)$*: on input the security parameter* λ, *this algorithm outputs a commitment key* ck.
- $c_m \leftarrow \mathsf{Com.Commit}(ck, m; r)$*: on input* ck, *a message* m *and a randomness* r, *it outputs a commitment* c_m.
- $b \leftarrow \mathsf{Com.Ver}(ck, c_m)$*: on input* ck *and* c_m, *it outputs a bit* $b \in \{0,1\}$.

For simplicity, we leave the randomness implicit and assume it is generated internally by Com.Commit. Next, we recall the zero-knowledge proof (ZKP) for a language $\mathcal{L}$. The ZKP allows a prover $\mathbb{P}$ to convince a verifier $\mathbb{V}$ that a series of instance x is in $\mathcal{L}$ without revealing anything other than this statement. Further, ZKP allows $\mathbb{P}$ to convince $\mathbb{V}$ with a witness

ω evidencing a fact that x is in $\mathcal{L}$, where $\mathcal{L}$ is defined by a relation predicate $\mathbf{P}_{\mathcal{L}(x,\omega)}$. Lyubashevsky et al. [75] proposed a non-interactive zero-knowledge proof (NIZK) system for a wide class of NP $\mathcal{L}$ over lattices.

Definition 11 (NIZK)**.** *Let $\mathcal{W}_{\mathcal{L}}(x)$ denote a generic set of witnesses such that for any $x \in \mathcal{L}$ and $\omega \in \mathcal{W}_{\mathcal{L}}(x)$, there is $\mathbf{P}_{\mathcal{L}(x,\omega)} = 1$. A NIZK includes the following three algorithms:*

- $\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$*: on input the security parameter λ, it outputs a common random string* crs*.*
- $\pi \leftarrow \mathsf{NIZK}.\mathbb{P}(crs, x, \omega)$*: on input* crs*, $x \in \mathcal{L}$ and $\omega \in \mathcal{W}_{\mathcal{L}}(x)$, this algorithm outputs a proof $\pi \in \{0,1\}^{poly(\lambda)}$.*
- $\{0,1\} \leftarrow \mathsf{NIZK}.\mathbb{V}(crs, x, \pi)$*: on input* crs*, x and π, it outputs 1 if $x \in \mathcal{L}$. Otherwise, it outputs 0.*

Definition 12 (NIZKAoK Security, [76])**.** *The algorithm $\Pi = (\mathsf{NIZK.Setup}, \mathsf{NIZK}.\mathbb{P}, \mathsf{NIZK}.\ \mathbb{V})$ is a NIZK for a relation R. The language $\mathcal{L}$ is defined by R if the following holds.*

- ***Completeness.*** *Let $x \in \mathcal{L}$. ω is the witness. If $\mathbf{P}_{\mathcal{L}(x,\omega)} = 1$, the following probability is negligible in λ:*

$$1 - Pr\left[1 \leftarrow \mathsf{NIZK}.\mathbb{V}(crs, x, \pi) \middle| \begin{matrix}\mathsf{crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda) \\ \pi \leftarrow \mathsf{NIZK}.\mathbb{P}(crs, x, \omega)\end{matrix}\right].$$

- ***Soundness.*** *For any $\mathbb{P}$, **Ext** is a PPT extractor. For any x and $\mathsf{pp} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$, the following is $\varepsilon(\lambda)$:*

$$Pr[\langle \mathbb{P}(\omega), \mathbb{V}\rangle(\mathsf{crs}, x) = 1 \wedge (x,\omega) \notin \mathcal{L} | \omega \leftarrow \textbf{\textit{Ext}}^{\mathbb{P}}(\mathsf{crs}, x)].$$

- ***Zero knowledge.*** *There is a PPT simulator **Sim** such that for any $\mathbb{V}^*$ with auxiliary information **aux**, $(x,\omega) \in \mathcal{L}$, it holds that: $View(\langle \mathbb{P}(w), \mathbb{V}^*\rangle(\mathsf{crs}, x, \textbf{\textit{aux}}))$ is computationally indistinguishable from $\textbf{\textit{Sim}}^{\mathbb{V}^*}(\mathsf{crs}, x, \textbf{\textit{aux}})$, where $\textbf{\textit{Ext}}^{\mathbb{P}}$ means that **Ext** has access to the entire process of $\mathbb{P}$, and $\textbf{\textit{Sim}}^{\mathbb{V}^*}$ means that **Sim** captures the randomness from a polynomial-size space of $\mathbb{V}^*$.*

G. Verifiable Partially OPRF

We first recall the framework of the lattice-based PRF proposed by Banerjee and Peikert [77], which serves as the foundation for constructing our VPOPRF. Let $\ell = \lceil \log_2 q \rceil$, $k \leftarrow R_q$ be the secret key, and let $\mathbf{a}_x \in R_q^{1\times\ell}$ denote the randomized encoding of the input $\mathbf{x} = (x_1, ..., x_L) \in \{0,1\}^L$.

Lemma 1 (PRF, [77])**.** *Let χ be the error distribution with parameter σ, and sample $k \leftarrow R_q$, the function $F_k(\mathbf{x}) = \lfloor \frac{p}{q} \cdot \mathbf{a}_\mathbf{x} \cdot k \rceil$ is a PRF over the Decision-$\mathsf{RLWE}_{n,q,\chi,m}$ if $q \gg p \cdot \sigma \cdot n \cdot \ell \cdot \sqrt{L}$, where L is the bit length of $\mathbf{x}$.*

Definition 13. *Let pp be the public parameter. A pseudorandom function PRF contains the following algorithms*

- $(k, pp) \leftarrow \mathsf{PRF.Setup}(1^\lambda)$ *is a probabilistic algorithm that outputs the public parameters pp and a PRF key k.*
- $\mathbf{a}_x \leftarrow \mathsf{PRF.Code}(pp, x)$ *is a deterministic algorithm that outputs $\mathbf{a}_x$ from secret input x.*
- $F_k(x) \leftarrow \mathsf{PRF.Eval}(pp, k, x)$ *is a deterministic algorithm that output a PRF value $F_k(x) = \lfloor \frac{p}{q} \cdot \mathbf{a}_x \cdot k \rceil$.*

We employ the instantiation of the PRF from Li et al. [78]. There is $\mathbf{a}_\mathbf{x} = a \cdot G(\mathbf{x})$, where $a \in R_q$, and the hash function $G : \{0,1\}^* \to R_q^{1\times\ell}$. According to the Theorem 4.1 of Yang et al. [79], $F_k(\mathbf{x}) = \lfloor \frac{p}{q} \cdot \mathbf{a}_\mathbf{x} \cdot k \rceil$ achieves weak pseudorandomness.

For a security parameter λ, output the public parameters pp and the master secret key msk. The functionality $\mathcal{F}_{\text{POPRF}}$ interacts with the servers S, an arbitrary user U, and an adversary $\mathcal{A}$. Set $\phi = 0$.
Key Generation:
S executes $\mathsf{F.KeyGen}$ to get the user exclusive key K for U.
Evaluation:
Let sid be the session identifier and pid be the party identifier.
On receiving $(\mathsf{F.Req}, \mathcal{I}, req, sid, pid)$ **from any** U:
- Record $(eval, pp, \mathcal{I}, req, sid, pid)$
- Send $(eval, pp, \mathcal{I}, req, sid, pid)$ to S.
On receiving $(\mathsf{F.Blind}, res, sid, pid)$ **from** S:
- Retrieve record $(eval, pp, res, sid, pid)$.
- If $(\mathsf{F.Blind}, res, sid, pid)$ has been received from S, set $\phi := \phi + 1$.
On receiving $(\mathsf{Query}, sid, pid)$ **from any** $\mathcal{A}$:
- Retrieve record $(eval, pp, res, sid, pid)$, if $\phi > 0$, set $\phi := \phi - 1$.
- If req is undefined, then pick $\rho \leftarrow \{0,1\}^\lambda$, and set $req := \rho$.
- Output $(\mathsf{Query}, pp, req, sid, pid)$ to U.

Fig. 2: The ideal functionality $\mathcal{F}_{POPRF}$.

Next, we propose a (partially) OPRF, followed by its verifiable extension, called VPOPRF. Our architecture mostly follows the framework of Tyagi et al. [46], while offering adapted security notions tailored to password-authenticated that ensure obliviousness and unpredictability for POPRFs.

Definition 14 (POPRF, [42])**.** *A POPRF F_K contains the following six probabilistic polynomial time (PPT) algorithms:*

- $(pp, msk) \leftarrow \mathsf{F.Setup}(1^\lambda)$ *generates the public parameters pp and the master secret key msk.*
- $st_K \leftarrow \mathsf{F.KeyGen}(pp, \mathcal{I}, msk)$ *generates the server state $st_K = (K, c_k, \rho_K)$ for the public input $\mathcal{I}$, where c_k is a commitment to K with opening randomness ρ_K.*
- $(st_U, req) \leftarrow \mathsf{F.Req}(pp, \mathcal{I}, x)$ *generates the user state st_U and a request req from the input $\mathcal{I}$ and x.*
- $res \leftarrow \mathsf{F.Blind}(pp, \mathcal{I}, req, st_K)$ *processes the request using the server state and returns a response res or $\perp$.*
- $Y \leftarrow \mathsf{F.Final}(pp, c_k, res, st_U)$ *verifies the response against the commitment c_k and returns the output Y.*
- $Y \leftarrow \mathsf{F.Eval}(\mathcal{I}, x, K)$*: is a deterministic algorithm that generates Y by taking as input K and $(\mathcal{I}, x)$.*

Note that F_K corresponds to a (plain) OPRF when the public message is fixed as $\mathcal{I} = \perp$. The ideal functionality $\mathcal{F}_{POPRF}$ is defined in Fig. 2. Let $\mathsf{negl}(\lambda)$ denote a negligible failure probability, we define the correctness, obliviousness and unpredictability for the POPRF F_K from Tyagi et al. [46].

Definition 15 (Correctness, [46])**.** *A POPRF $\mathcal{F}_K$ is correct if the following probability is $1 - \mathsf{negl}(\lambda)$:*

$$\Pr\left[Y = \mathsf{F.Eval}(\mathcal{I}, x, K) \middle| \begin{matrix}(pp, msk) \leftarrow \mathsf{F.Setup}(1^\lambda) \\ st_K \leftarrow \mathsf{F.KeyGen}(pp, \mathcal{I}, msk) \\ (st_U, req) \leftarrow \mathsf{F.Req}(pp, \mathcal{I}, x) \\ res \leftarrow \mathsf{F.Blind}(pp, \mathcal{I}, req, st_K) \\ Y \leftarrow \mathsf{F.Eval}(\mathcal{I}, x, K)\end{matrix}\right]$$

Definition 16 (Obliviousness, [46], [42])**.** *A POPRF F_K is oblivious if for all PPT adversaries $\mathcal{A}$, there is a PPT simulator $\mathcal{S}$ such that the following probability is $\mathsf{negl}(\lambda)$*

$$\mathrm{Adv}^{\mathsf{OBL}}_{\mathcal{A},\mathcal{S}}(\lambda) = \left|\Pr\left[\mathsf{OBL}^{\mathcal{A},1}_{K,\mathcal{S}}(\lambda) = 1\right] - \Pr\left[\mathsf{OBL}^{\mathcal{A},0}_{K,\mathcal{S}}(\lambda) = 1\right]\right|.$$

Definition 17 (Unpredictability, [46], [42])**.** *A POPRF $\mathcal{F}$ has unpredictability against malicious client if for all PPT $\mathcal{A}$ the following advantage probability is $\mathsf{negl}(\lambda)$*

$$\mathrm{Adv}^{\mathsf{UNP}}_{\mathcal{A},\mathcal{S}}(\lambda) = \left|\Pr\left[\mathsf{UNP}^{\mathcal{A},1}_{K,\mathcal{S}}(\lambda) = 1\right] - \Pr\left[\mathsf{UNP}^{\mathcal{A},0}_{K,\mathcal{S}}(\lambda) = 1\right]\right|.$$

$\underline{\mathsf{F.Setup}(1^\lambda)}$
1: $ck_i \leftarrow \mathsf{Com.Setup}(1^\lambda)$, $i \in \{0,1\}$
2: $crs_i \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$, $i \in \{0,1\}$
3: $a \leftarrow R_q$, $msk \leftarrow R_q$
4: $pp := (ck_0, ck_1, crs_0, crs_1, a)$
5: **return** (pp, msk)

$\underline{\mathsf{F.Req}(pp, \mathcal{I}, x)}$
1: $r \leftarrow \mathcal{X}_\sigma$, $\mathbf{s}, \mathbf{e} \leftarrow \mathcal{X}_\sigma^{1\times\ell}$
2: $c_r \leftarrow \mathsf{Com.Commit}(ck_0, r; \rho_r)$
3: $\hat{x} := \mathsf{Code}(\mathcal{I}\|x)$, $\mathbf{a}_{\mathcal{I},x} := a \cdot G(\hat{x})$
4: $\mathbf{a} := H_1(\mathcal{I}, c_r)$
5: $\mathbf{b} := \mathbf{a}\cdot\mathbf{s} + \mathbf{e} + \mathbf{a}_{\mathcal{I},x}$
6: $\pi_0 \leftarrow \mathsf{NIZK.P}(crs_0, X_U, w_U)$
7: $req := (c_r, \mathbf{b}, \pi_0, \mathcal{I})$
8: $st_U := (\mathbf{s}, x, req)$; **return** (st_U, req)

$\underline{G(\hat{x})}$
1: **if** $G(\hat{x}) = \perp$ **then**
2: $G(\hat{x}) \leftarrow R_q^{1\times\ell}$
3: **return** $G(\hat{x})$

$\underline{\mathsf{F.KeyGen}(pp, \mathcal{I}, msk)}$
1: $K \leftarrow H_0(\mathcal{I}, msk)$
2: $c_k \leftarrow \mathsf{Com.Commit}(ck_1, K; \rho_K)$
3: **return** $st_K := (K, c_k, \rho_K)$

$\underline{\mathsf{F.Blind}(pp, \mathcal{I}, req, st_K)}$
1: $\mathbf{a} := H_1(\mathcal{I}, c_r)$
2: **if** $\mathsf{NIZK.V}(crs_0, X_U, \pi_0) \neq 1$
3: **return** $\perp$
4: **else** $\mathbf{e}_0, \mathbf{e}_1 \leftarrow \mathcal{X}_\sigma^{1\times\ell}$
5: $PK := \mathbf{a}\cdot K + \mathbf{e}_0$
6: $\mathbf{c}_1 := \mathbf{b}\cdot K + \mathbf{e}_1$
7: $\pi_1 \leftarrow \mathsf{NIZK.P}(crs_1, X_S, w_S)$
8: **return** $res := (PK, \mathbf{c}_1, \pi_1)$

$\underline{\mathsf{F.Final}(pp, c_k, res, st_U)}$
1: **if** $\mathsf{NIZK.V}(crs_1, X_S, \pi_1) \neq 1$
2: **return** $\perp$
3: **else** $\mathbf{y} := \lfloor \mathbf{c}_1 - PK\cdot\mathbf{s} \rceil_p$
4: $z := H_2(\mathbf{y}, x, \mathcal{I})$
5: **return** z

$\underline{\mathsf{F.Eval}(pp, msk, \mathcal{I}, x)}$
1: $K \leftarrow H_0(\mathcal{I}, msk)$
2: $\hat{x} := \mathsf{Code}(\mathcal{I}\|x)$
3: $\mathbf{a}_{\mathcal{I},x} := a \cdot G(\hat{x})$
4: $\mathbf{y} := \lfloor \mathbf{a}_{\mathcal{I},x} \cdot K \rceil_p$
5: $z := H_2(\mathbf{y}, x, \mathcal{I})$
6: **return** z

$\underline{H_0(\mathcal{I}, msk)}$
1: **if** $H_0(\mathcal{I}, msk) = \perp$ **then**
2: $H_0(\mathcal{I}, msk) \leftarrow \mathcal{X}_\sigma$
3: **return** $H_0(\mathcal{I}, msk)$

$\underline{H_1(\mathcal{I}, c_r)}$
1: **if** $H_1(\mathcal{I}, c_r) = \perp$ **then**
2: $H_1(\mathcal{I}, c_r) \leftarrow R_q^{1\times\ell}$
3: **return** $H_1(\mathcal{I}, c_r)$

$\underline{H_2(\mathbf{y}, x, \mathcal{I})}$
1: **if** $H_2(\mathbf{y}, x, \mathcal{I}) = \perp$ **then**
2: $H_2(\mathbf{y}, x, \mathcal{I}) \leftarrow \{0,1\}^\lambda$
3: **return** $H_2(\mathbf{y}, x, \mathcal{I})$

The user-side proof $\mathbb{P}_U$
$X_U = (c_r, \mathbf{b}, \mathbf{a}, a, ck_0, \mathcal{I}, \beta_r, \beta_s, \beta_e)$
$w_U = (r, x, \mathbf{s}, \mathbf{e}, \rho_r)$
$\mathbb{P}_U = \{X_U \mid \exists w_U :\ \|r\|_\infty \le \beta_r$
$\wedge\ \|\mathbf{s}\|_\infty \le \beta_s\ \wedge\ \|\mathbf{e}\|_\infty \le \beta_e$
$\wedge\ \mathbf{a} = H_1(\mathcal{I}, c_r)$
$\wedge\ c_r = \mathsf{Com.Commit}(ck_0, r; \rho_r)$
$\wedge\ \mathbf{b} = \mathbf{a}\cdot\mathbf{s} + \mathbf{e} + a\cdot G(\mathsf{Code}(\mathcal{I}\|x))\}$

The server-side proof $\mathbb{P}_S$
$X_S = (\mathbf{c}_1, \mathbf{b}, c_k, PK, ck_1, \mathbf{a}, c_r, \mathcal{I}, \beta_K, \beta_e)$
$w_S = (K, \mathbf{e}_0, \mathbf{e}_1, \rho_K)$
$\mathbb{P}_S = \{X_S \mid \exists w_S :\ \|K\|_\infty \le \beta_K$
$\wedge\ \|\mathbf{e}_0\|_\infty \le \beta_e\ \wedge\ \|\mathbf{e}_1\|_\infty \le \beta_e$
$\wedge\ \mathbf{a} = H_1(\mathcal{I}, c_r)$
$\wedge\ c_k = \mathsf{Com.Commit}(ck_1, K; \rho_K)$
$\wedge\ PK = \mathbf{a}\cdot K + \mathbf{e}_0$
$\wedge\ \mathbf{c}_1 = \mathbf{b}\cdot K + \mathbf{e}_1\}$

Fig. 3: The details of our VPOPRF construction are shown on the left, while the specifics of $\mathbb{P}_U$ and $\mathbb{P}_S$ are presented on the right. Here, ρ_r and ρ_k represent the randomness of the commitment, and β_s, β_r, β_k, and β_e denote the respective bounds. Products of two vectors denote componentwise multiplication over R_q. All algorithms unpack pp and their tuple inputs before use. Inconsistent copies of $\mathcal{I}$ are rejected. The public statements X_U and X_S are reconstructed from the current inputs and fixed public parameters, with $\mathbf{a} = H_1(\mathcal{I}, c_r)$. Commitment randomness ρ_r and ρ_K is sampled freshly according to the commitment scheme. The states st_U and st_K are maintained locally by the user and server, respectively; c_k is supplied to $\mathsf{F.Final}$ as the reference commitment. The final OPRF output supplied to qaPAKE is $Y_u = z$.

Let Code convert its input into a binary representation, we realize the 2HashDH OPRF idea [51] over lattices, and propose a lightweight VPOPRF as shown in Fig. 3. In contrast to Albrecht et al. [44], [80], whose $\mathsf{F.Req}$ computation relies on a non-linear binary decomposition of inputs, our construction employs a weak PRF [79] with a linear input representation that integrates efficiently with zero-knowledge proofs, making it more suitable for lightweight password-based authentication. Notably, our modification only affects the input-dependent term $\mathbf{a}_{\mathcal{I}|x}$, and does not alter the underlying 2HashDH OPRF structure of Albrecht et al. [44], [80]. Hence, our VPOPRF maintains the round-optimal, and supports obliviousness and unpredictability in Definition 16 and Definition 17.

III. Scheme Architecture and Security Model

A. System Model

We present a quantum-safe anonymous PAKE, named qaPAKE. Users can log in to the server anonymously using their username and password, and establish a secure channel (generating temporary high entropy session keys) without device restrictions. Our qaPAKE includes two participants.

User. The user holds a username and a password. Users need to register with the server and log in with the correct username and password. After mutual authentication, users can establish a high entropy session key for a secure channel.

Server. The server holds a credential dataset $\mathcal{T}$. During the offline phase, the server can perform pre-computation on $\mathcal{T}$ with a fresh key. In the online phase, the server interacts with users and establishes a high entropy session key.

Password Dictionary. Each $U \in \mathbf{User}$ holds a username and a password $pw \in \mathcal{D}$, where $\mathcal{D}$ is the password space with the size $|\mathcal{D}|$. It has been shown that passwords follow the CDF-Zipf distribution [81], [69], [82]. Notably, the CDF-Zipf based formulation $C' \cdot q_{send}^{s'}(\lambda) + \varepsilon(\lambda)$ well approximates the real attacker's $Adv : q_{send} \in [1, |\mathcal{D}|]$, where C' and s' denote the Zipf parameters (e.g., $C' = 0.037433$, $s' = 0.187227$ for 32 million Rockyou passwords [81]). This CDF-Zipf based formulation is more accurate than other formulations used, e.g., the min-entropy model [83]. Therefore, we recommend using the accurate Zipf-based formulation for qaPAKE.

Let U^i and S^j denote protocol instances of $U \in \mathbf{User}$ and $S \in \mathbf{Server}$, respectively. Let I denote either a user instance or a server instance. In qaPAKE, the server performs batch-level offline precomputation using a fresh signing key pair (sk_s, pk_s). The OPRF key K is maintained across broadcast batches, whereas RID_u and RID_s are generated afresh for each online session. During the online phase, the user and server execute the authentication and key-establishment procedure in Fig. 4. In addition, qaPAKE allows users to revoke credential files in $\mathcal{T}$. The combination of anonymous revocation and registration allows users to update their passwords.

B. Security Model

In this section, we introduce the authentication security model. Note that, security requirements under traditional settings (non quantum computers) are still necessary [84], [59]. To formally capture the capabilities of an adversary $\mathcal{A}$ in our protocol, and specify how the adversary interacts with honest parties, we employ the *Bellare-Pointcheval-Rogaway* (BPR) model [85], which is by far the most popular model for password authentication protocol (e.g.,[86], [87]). Each $U \in \mathbf{User}$ holds a username and a password pw. $S \in \mathbf{Server}$ holds the set of registered credentials.

Adversarial model. To simulate aPAKE protocols more realistically, we need to clarify the capabilities of an adversary before defining the security model. We consider $\mathcal{A}$ with quantum computing capabilities to control the external network fully. $\mathcal{A}$ is free to manipulate messages and adaptively requests any session keys. Furthermore, according to the adversarial model of BPR [85] and a series of password-based protocols [36], [88], we summarize the capacities of $\mathcal{A}$ for qaPAKE in Table III. Each capability is realistic and reasonable.

Queries. The adversary $\mathcal{A}$ interacts with the protocol participants using oracle queries that simulate $\mathcal{A}$'s capabilities in a real attack. The query models available to $\mathcal{A}$ are as follows.

- Execute(U^i, S^j) captures a passive attack, such as eavesdropping. The output of Execute consists of the messages exchanged during the honest execution.
- Send(I, m) captures an active attack. The adversary sends m to instance I and receives the response produced by I according to the protocol.
- Test(I) defines the semantic security of the session key and may be invoked at most once on a fresh instance I. The challenger samples a secret bit $b \leftarrow \{0,1\}$. If $b = 1$, it returns the session key SK_I accepted by I. Otherwise, it returns an independently and uniformly sampled string from $\{0,1\}^{|SK_I|}$. The bit b is not revealed to $\mathcal{A}$.
- Reveal(I) returns the session key accepted by I.
- Reveal$^+$(I) captures the honest-but-curious server. $\mathcal{A}$ obtains the server key K generated in the offline phase.
- Corrupt(I) captures the corrupt attack. If $I = U$, it outputs the password pw_u of U, and can download st_0 from CS. If $I = S$, it outputs the st_i of S_i.

Partnering. Let sid be the session identifier. The value $pid = H_3(RID_u \| RID_s)$ binds the two session pseudonyms and is not a partner identity. Instances U^i and S^j are partnered if: 1) both instances have accepted; 2) $sid_{U^i} = sid_{S^j}$; and 3) their local values of (pid, ek_u, ct_s, pk_s) agree, with the instances executing the user and server roles, respectively. These conditions are evaluated by the security experiment and do not require the server to learn the user's identity.

Freshness. We define I as fresh if the following conditions are true: 1) I has accepted and computed a session key. 2) Neither I nor its partner has been asked for a query Reveal(I).

Semantic Security. In the sequence of games, $\mathcal{A}$ can ask a polynomial number of querys Execute(U^i, S^j), Send(I, m), and Reveal(I). Finally, $\mathcal{A}$ issues $\mathsf{Test}(I)$ and outputs a guess $b' \in \{0,1\}$ for the secret challenge bit b. For any PPT $\mathcal{A}$, the advantage holds that $Adv_{\mathcal{A}}^{ake} = 2Pr[b' = b] - 1$. A lattice-based password-only aPAKE $\mathcal{P}$ achieves the semantic security if the advantage $Adv_{\mathcal{A}}^{ake}$ is negligible.

Quantum resistance. The most influential quantum attack algorithms are Shor's [31] and Grover's [89]. Quantum computers can efficiently solve integer factorization and discrete logarithm problems with Shor's algorithm. Grover's algorithm allows quantum computers to speed up the search for unstructured databases and hash collisions. To avoid the effects caused by both algorithms above, we construct our protocol based on LWE [70], and the hash function such as SHAKE128 [37], which could offer 128 bits of post-quantum security against collisions and preimage attacks.

C. Anonymity

The security model mentioned above is widely used to characterize the standard PAKE protocol [90], [91], [92], [36]. Notably, all participants are publicly identifiable, and each username corresponds to pw (registered between the honest U and S). However, in the aPAKE, the direct correspondence between ID and pw is implicit, indicating that it is unknown whether pw is registered between U and S. Hence, when characterizing protocol anonymity, it is necessary to consider: 1) U is anonymous to S that saves the credential cre. 2) S's database $\mathcal{T}_S$ is dynamic, and $cre \in \mathcal{T}$ are allowed to be leaked, i.e., $\mathcal{T}_{\mathcal{A}}$ contains valid cre. 3) $\mathcal{A}$ can use different $\mathcal{T}$ to reveal U identities, rather than just using $\mathcal{T}_S$. In other words, anonymity requires that the external $\mathcal{A}$ cannot determine whether both parties have generated the same session key.

TABLE III: Capabilities of the adversary. †

Index	Description
C-1	$\mathcal{A}$ can offline enumerate the Cartesian product $D_{id} \times D_{pw}$
C-2	$\mathcal{A}$ can determine the identity ID_i of victim.
C-3	$\mathcal{A}$ fully controls the communication channel.
C-4	$\mathcal{A}$ can learn the password pw_i of victim.
C-5	$\mathcal{A}$ can learn the previous session key(s).
C-6	$\mathcal{A}$ can learn the server's long-term secret key and all other data stored in $\mathcal{T}$ only upon server corruption (forward secrecy).

† $\mathcal{A}$ denotes the adversary; $\mathcal{T}$ is the server database. D_{id} denotes the identity space; D_{pw} denotes the password space.

We employ the anonymity model of Viet et al. [17] to characterize the security of our qaPAKE. At a high level, the anonymous model allows $\mathcal{A}$ to learn whether two parties hold cre by Execute-query and Reveal-query. In addition, $\mathcal{A}$ can learn the registered credentials by Corrupt-query.

Definition 18 ([17])**.** *Let pw_0, pw_1 be the two challenge passwords, and cre_i be the credential. $\mathcal{T}_0$ and $\mathcal{T}_1$ are two credential sets. When $\mathcal{A}$ holds cre_0 and cre_1, a server* $\mathrm{S} \in \mathbf{Server}$ *with $\mathcal{T}_S$ is distinguishing if $|\mathcal{T}_S \cap \{cre_0, cre_1\}| = 1$. When $\mathcal{A}$ holds $\mathcal{T}_0$ and $\mathcal{T}_1$, $U \in$* **User** *holds pw_u and the corresponding cre_u is distinguishing if $cre_u \in \mathcal{T}_0 \Delta \mathcal{T}_1$, where Δ denotes the symmetric difference between the two sets.*

User Anonymity. Let Ch be a challenger. $\mathcal{A}$ picks $pw_0, pw_1 \leftarrow \mathcal{D}$ and computes $cre_0, cre_1 \in \mathcal{T}_{\mathcal{A}}$. $\mathcal{A}$ sends cre_0, cre_1 to Ch. Then, Ch samples a secret bit $b \leftarrow \{0,1\}$, selects cre_b, and acts as U_{Ch}. $\mathcal{A}$ interacts with each participant via $\mathsf{Execute}, \mathsf{Reveal}, \mathsf{Reveal}^+, \mathsf{Corrupt}$. $\mathcal{A}$ must adhere to the following conditions: 1) $\mathcal{A}$ never called $\mathsf{Corrupt}$ (U_{Ch}). 2) $\mathcal{A}$ never called $\mathsf{Reveal}(U_{\mathsf{Ch}}^l)$ or $\mathsf{Reveal}(S_j^t)$ after $\mathsf{Execute}$ $(U_{\mathsf{Ch}}^l, S_j^t)$ where S_j is a distinguishing server. 3) $\mathcal{A}$ never called Reveal^+ (S_j^t) and $\mathsf{Execute}$ $(U_{\mathsf{Ch}}^l, S_j^t)$ for some *l, t* where S_j is a distinguishing server. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.

For any PPT $\mathcal{A}$, the advantage of disrupting user anonymity holds that $Adv_{\mathcal{A}}^{ua} = 2Pr[b' = b] - 1$. Our qaPAKE protocol $\mathcal{P}$ achieves user anonymity if $Adv_{\mathcal{A}}^{ua}$ is negligible.

Server Anonymity. Generally, server anonymity is not considered in the aPAKEs [13], [14], [15], [17], [18], [19], [20], [21]. However, in practice, $\mathcal{T}_S$ will change over time, and server anonymity is the key to characterizing credential database information leakage [9]. Specifically, $\mathcal{A}$ chooses two distinct credential sets $\mathcal{T}_0, \mathcal{T}_1$, where $|\mathcal{T}_0| = |\mathcal{T}_1|$. $\mathcal{A}$ sends $\mathcal{T}_0, \mathcal{T}_1$ to Ch. Then, Ch picks $\mathcal{T}_b$ and acts as S_{Ch}, where $b \in \{0,1\}$. $\mathcal{A}$ interacts with each participant via $\mathsf{Execute}, \mathsf{Reveal}, \text{and } \mathsf{Corrupt}$. $\mathcal{A}$ cannot achieve the following two cases: 1) $\mathcal{A}$ never called $\mathsf{Corrupt}$ (S_{Ch}); 2) $\mathcal{A}$ never called Reveal (S_{Ch}^l) or Reveal (U_j^t) after $\mathsf{Execute}$ $(U_j^t, S_{\mathsf{Ch}}^l)$ where U_j is a distinguishing user. For any PPT $\mathcal{A}$, the advantage of disrupting server anonymity holds that $Adv_{\mathcal{A}}^{sa} = 2Pr[b' = b] - 1$. Our protocol $\mathcal{P}$ achieves

server anonymity if $Adv^{sa}_{\mathcal{A}}$ is negligible.

IV. OUR NEW SCHEME

In this section, we propose qaPAKE, a quantum-safe password-only anonymous PAKE over lattices as shown in Fig. 4. Our qaPAKE operates in a user-server setting. The server S holds an OPRF key K and maintains an authentication-file set $\mathcal{T} = \{File[i]\}_{i\in[n]}$, where $File[i] = (l_i, pk_i)$ and $l_i = H_2(pk_i)$. A user U_i maintains only a username I_i and a password pw_i and reconstructs the corresponding credential key pair (sk_i, pk_i) from the OPRF output during each login.

Following prior aPAKE constructions [17], [18], [9], qaPAKE separates batch-level offline precomputation from session-specific online authentication. For each broadcast batch, S generates a fresh signing key pair (sk_s, pk_s) and encrypts the verification key pk_s under the credential-derived symmetric keys $k_i = H_1(pk_i)$. The resulting ciphertext vector C and label vector L are precomputed for distribution during login. In the online phase, U_i reconstructs the credential material, recovers pk_s, and participates in anonymous membership authentication and session-key establishment with S.

Full-download vs. pseudonym-based lookup. In conventional pseudonym-based database lookup [14], [19], [18], the server resolves a submitted pseudonym to a particular authentication record. Although the pseudonym may conceal the user's explicit identity, this retrieval process reveals which record is involved in the authentication attempt. Under the mutual-anonymity notion of Vasco et al. [9], user anonymity requires authentication without revealing the association between a session and a particular authentication file, whereas server anonymity concerns the privacy of the server's authentication-file set $\mathcal{T}$. In qaPAKE, downloading a masked representation of $\mathcal{T}$ allows the user to select the corresponding entry locally without disclosing that selection to the server. Protection of the authentication-file set additionally relies on OPRF-derived credential masking and ciphertext confidentiality.

Our qaPAKE does not require users to maintain a location index identifying the position of $File[i]$ within $\mathcal{T}$. Index-based oblivious-transfer constructions that require the user to supply such a position [18], [19], [20], [21] therefore require an additional mechanism for recovering that position under our password-only requirement. In our qaPAKE, U_i reconstructs pk_i from the OPRF output using only its username and password, computes $l_i = H_2(pk_i)$, and locally searches for a matching entry in $L = (l_1, \ldots, l_n)$. The matching position identifies the corresponding ciphertext in the aligned vector $C = (c_1, \ldots, c_n)$. Neither the selected label nor its position is submitted to the server as a lookup request, and the user need not maintain either between logins. Reconstructing a pseudonym at each login could also avoid persistent client-side storage, but submitting it for server-side record lookup would still reveal the corresponding authentication record.

This design incurs $O(n)$ server-to-client communication because the user receives the complete vectors (L, C), where $n = |\mathcal{T}|$. Conventional pseudonym-based lookup avoids this full download and is therefore more communication-efficient when revealing the selected record to the server is acceptable. The advantage of qaPAKE's full-download design is the combination of password-only authentication and undisclosed local record selection, obtained at the communication cost.

Design idea. We employ the database-masking idea of Viet et al. [17]. In qaPAKE, S encrypts the same batch-specific verification key pk_s under $k_i = H_1(pk_i)$ associated with each authentication file. A user entering the registered username and password reconstructs the corresponding pk_i, derives the same symmetric key, and decrypts the aligned ciphertext. The server stores OPRF-derived credential material rather than plaintext passwords. The recovered pk_s is used in the user's authentication tag, the verification of the server's signature, and the derivation of the session key and server confirmation tag. Importantly, pk_s is not the shared secret established by the KEM: each login uses a fresh KEM key pair and a fresh encapsulation to obtain session-specific key material.

The capabilities C-1 and C-6 in Section III-B distinguish offline enumeration from server compromise. If an authentication file directly stores $p_i = H_0(I_i \| pw_i)$, an adversary can precompute candidate values over $D_{id} \times D_{pw}$ without knowing any server-held secret and compare them with the stored files after compromise. In qaPAKE, the mapping from the user's credentials to the stored file additionally depends on K. Specifically, the user obtains $Y_i = F_K(p_i)$ and deterministically derives $(sk_i, pk_i) \leftarrow \mathsf{EncKG}(1^\lambda, Y_i)$, using the seeded key-generation interface in Definition 6. The server stores pk_i and $l_i = H_2(pk_i)$ rather than p_i or pw_i.

Under the security of the OPRF, an adversary without K cannot derive unqueried OPRF outputs merely by hashing candidate credentials. This does not imply resistance to offline dictionary attacks after both K and $\mathcal{T}$ are compromised. In that case, the adversary can evaluate F_K on candidate inputs, derive the corresponding credential public keys, and compare them with the compromised authentication files.

DPaSE of Das et al. [93] addresses distributed password-authenticated symmetric-key encryption. Our setting instead concerns anonymous authentication and fresh session-key establishment between a user and a server. Accordingly, qaPAKE uses symmetric encryption to distribute the batch-specific verification key and a KEM to establish the session-specific shared secret. The credential key-generation algorithm EncKG is distinct from the symmetric encryption algorithms SEnc and SDec used for the broadcast ciphertexts. In particular, the broadcast decryption key is $H_1(pk_i)$, not sk_i.

For clarity, Fig. 4 presents the construction in the honest-but-curious setting. The hash functions $\{H_0, \ldots, H_7\}$ are domain-separated from one another and from the hash functions internal to the OPRF subprotocol, and all concatenations use unambiguous encodings. The output of H_1 has the length required by the symmetric encryption scheme. Unless shown explicitly, any randomness required by SEnc is generated internally according to its specification.

Password registration. Before registration, S initializes and maintains K according to the OPRF key-generation procedure. Over a secure channel, U_i computes $p_i = H_0(I_i|pw_i)$, where $pw_i \in \mathcal{D}$ and $I_i \in \{0,1\}^\ell$, and executes the OPRF in Fig. 3 with S, using private inputs p_i and K, respectively, and public input $\mathcal{I} = \perp$. From the output $Y_i = F_K(p_i)$, U_i deterministically derives $(sk_i, pk_i) \leftarrow \mathsf{EncKG}(1^\lambda, Y_i)$ and sends pk_i to S

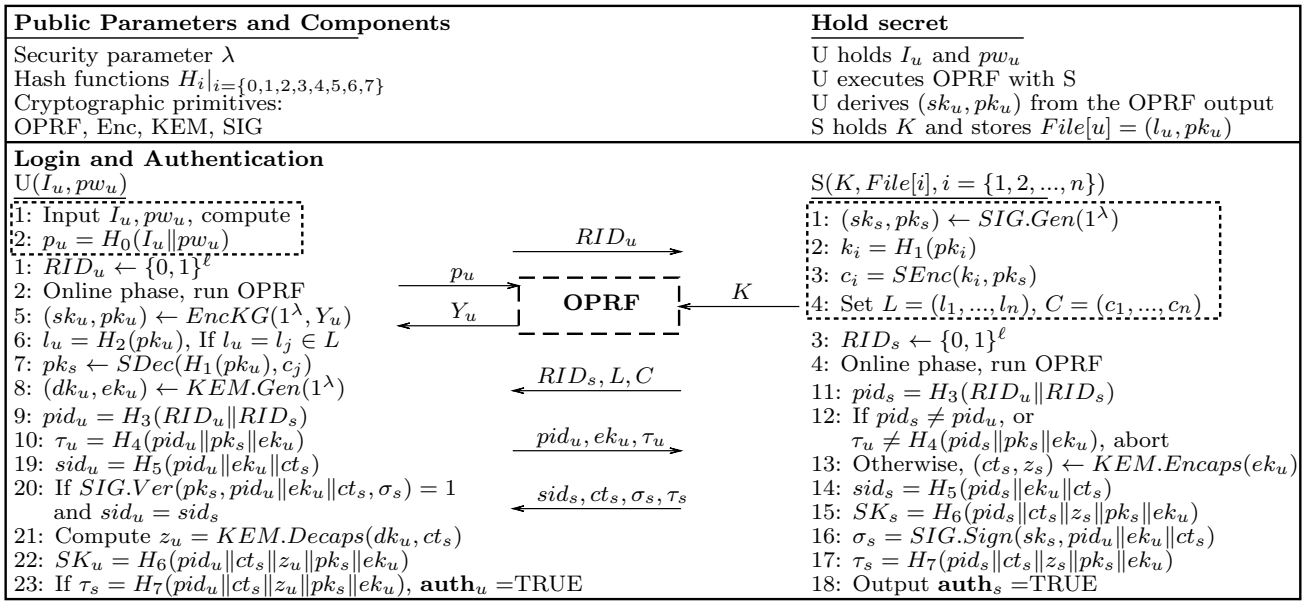


Fig. 4: Detailed workflow of qaPAKE. For clarity, the protocol is illustrated in the honest-but-curious setting, and the offline computations are enclosed by dotted boxes. The OPRF subprotocol is instantiated as shown in Fig. 3 with $\mathcal{I} = \perp$. (sk_s, pk_s) is an ephemeral key pair that is refreshed for each broadcast batch. In the malicious-server setting, to prevent S from compromising user anonymity by embedding different pk_s in different c_i, S additionally generates a NIZK proof, as defined in Definition 11, that all c_i encrypt the same pk_s. Notably, the proof is generated during the offline phase, while the user verifies it during login in about 4.3 ms on a common consumer smartphone.

through the secure channel. The server computes $l_i = H_2(pk_i)$ and stores $File[i] = (l_i, pk_i)$ in $\mathcal{T}$. The user need not maintain the key pair or its database position between logins.

Login. Fig. 4 specifies the computations and message exchanges. The OPRF key K is maintained across batches, whereas a fresh signing key pair (sk_s, pk_s) and aligned vectors (L, C) are prepared for each broadcast batch. The server maintains the selected batch state throughout each login. The pseudonyms RID_u, RID_s, the user's KEM key pair, and the server's encapsulation are generated afresh per session. The KEM shared secret supplies session-specific key material, and the domain-separated functions H_6 and H_7 derive the session key and server confirmation tag from this secret and the session context, rather than deriving the tag from the final session key. The user terminates without accepting a key if no label matches or decryption fails, and accepts only after the session identifier, server signature, and confirmation tag are verified.

Protection against malicious parties. The malicious-party extension requires additional proofs for OPRF message validity and broadcast consistency. The proof-based validation of lattice-based relations [79], [44], [42] is used to specify the required checks. In the OPRF component, S must verify the user-side proof before evaluating a request with its long-term key K, and the user must verify the server-side proof before using the resulting OPRF output. These checks enforce the request and response relations specified in Fig. 3; their guarantees depend on the validity of those relations and the security of the instantiated proof system.

A malicious server may additionally attempt to distinguish users by encrypting different verification keys $pk_s[i]$ in ciphertexts associated with different authentication files. Since τ_u depends on the recovered verification key, the server could test the values $pk_s[i]$ against the received τ_u and identify which ciphertext the user decrypted. The malicious-server extension therefore requires a NIZK proof π_{eq}, using the interface in Definition 11, for consistency of the broadcast ciphertexts.

More precisely, the public statement is (L, C). The witness consists of a common verification key pk_s and credential public keys $\{pk_i\}_{i\in[n]}$, satisfying $\bigwedge_{i=1}^{n} (l_i = H_2(pk_i) \ \wedge \ c_i = \mathsf{SEnc}(H_1(pk_i), pk_s))$. Using a NIZK system for this relation, the server generates π_{eq} during batch-level offline precomputation and sends it together with (RID_s, L, C). The user verifies the proof for the received (L, C) before decrypting the selected ciphertext and terminates if verification fails. The proof and its verification are omitted from the honest-but-curious presentation in Fig. 4.

Lemma 2 (Correctness)**.** *Let $\mathcal{T}$ be the set of registered authentication files stored by S. With input $p_i = H_0(I_i \| pw_i)$, U_i executes the VPOPRF with S and derives (sk_i, pk_i) from its output. If $File[i] = (l_i, pk_i) \in \mathcal{T}$ is included in the broadcast batch without a label collision, honest execution with unmodified message delivery yields $SK_u = SK_s$.*

Proof: With the registered input p_i and the same OPRF key K, VPOPRF correctness gives $Y_u = Y_i = F_K(p_i)$. Deterministic key generation therefore yields $(sk_u, pk_u) = \mathsf{EncKG}(1^\lambda, Y_u) = (sk_i, pk_i)$. Thus, $l_u = H_2(pk_u) = l_i$, and U_i selects the aligned ciphertext c_i. By the correctness of symmetric encryption in Definition 9, $\mathsf{SDec}(H_1(pk_u), c_i) = pk_s$.

Both parties use the same RID_u and RID_s, so $pid_u = H_3(RID_u \| RID_s) = pid_s$, and the server's verification of τ_u succeeds. The server executes $(ct_s, z_s) \leftarrow \mathsf{KEM.Encaps}(ek_u)$. The user obtains the same session identifier $sid_u = H_5(pid_u \| ek_u \| ct_s) = sid_s$, and the signature σ_s verifies under the recovered pk_s. By KEM correctness, $z_u = \mathsf{KEM.Decaps}(dk_u, ct_s) = z_s$. There is $SK_u = H_6(pid_u \| ct_s \| z_u \| pk_s \| ek_u) = H_6(pid_s \| ct_s \| z_s \| pk_s \| ek_u) = SK_s$. These equalities imply $H_7(pid_u \| ct_s \| z_u \| pk_s \| ek_u) =$

τ_s, so the user's final verification succeeds. □

V. Security Analysis

Our qaPAKE is provably secure in the formal model defined in Section III-B, under the assumption that the hash function closely behaves like a random oracle and that the Decision-$\mathsf{RLWE}_{n,q,\chi,m}$ [70] is intractable. Before detailing our security proof, we first point out two important observations.

On the one hand, the first observation concerns the use of the random oracle. Bellare and Rogaway [94] introduced the random oracle model (ROM), which provides each party with access to an oracle that implements a truly random function. In security proofs, it is common to model a hash function through such an oracle. In quantum settings, however, an adversary equipped with a quantum computer can implement the public hash function and evaluate it in superposition. Boneh et al. [95] proposed the quantum random oracle model (QROM) to capture this capability. It is important to note that QROM is not the only framework for proving quantum security [84]. We first demonstrate the security and anonymity of our qaPAKE through the ROM in Section V. Then, we discuss the quantum security of our qaPAKE in Section V-A.

On the other hand, our construction employs three lattice-based cryptographic primitives, namely oblivious pseudorandom function (OPRF), lattice-based signatures, and lattice-based encryption (denoted as Enc). We reduce the security of the OPRF to Decision-$\mathsf{RLWE}_{n,q,\chi,m}$, and satisfy unpredictability and obliviousness. Unpredictability and obliviousness mean that the value of the PRF is independent of the input secret. Specifically, obliviousness ensures that the adversary cannot predict the output of OPRF through user-side input. Unpredictability ensures that the server can learn nothing except its inputs. For clarity and convenience, we denote the $\mathcal{A}$'s advantage in breaking the OPRF by $Adv_{\mathcal{A}}^{OPRF}(\lambda)$. In addition, we denote the $\mathcal{A}$'s advantage in breaking Enc by $Adv_{\mathcal{A}}^{Enc}(\lambda)$.

Theorem 1. *Let $\mathcal{P}$ be the proposed qaPAKE in Fig. 4. Let $\mathcal{D}$ be the password space that follows Zipf's law [81]. ℓ is the length of an ephemeral identity. For a PPT adversary $\mathcal{A}$, $\mathcal{A}$ can ask at most q_{send} Send-queries, q_{exe} Execute-queries, and q_h random oracle queries. We have*

$$\begin{aligned} Adv_{\mathcal{A}}^{ake}(\lambda) \le\ & 2C' q_{send}^{s'} + 2Adv_{\mathcal{A}}^{OPRF}(\lambda) + 2Adv_{\mathcal{A}}^{Enc}(\lambda) \\ & + 2N_B Adv_{\mathcal{A}}^{\mathsf{SIG}}(\lambda) + 2N_U Adv_{\mathcal{A}}^{\mathsf{KEM}}(\lambda) + \frac{q_h^2}{2^{\kappa_H}} \\ & + \frac{(2q_{exe}+q_{send})^2}{2^{\ell}} + \frac{2q_{send}}{2^{|\tau_u|}} + \frac{2N_U q_h}{2^{\kappa_z}}. \end{aligned}$$

where C' and s' denote the Zipf parameters ($C' = 0.037433$, $s' = 0.187227$ for 32 million Rockyou passwords [81]).

Proof: The proof is set up in terms of a sequence of games starting from the real case Game_0 and ends at the ideal case Game_9. For each Game_n $(n = 0, 1, ..., 9)$, we define the following events: 1) Succ_n: $\mathcal{A}$ correctly guesses the bit b involved in the Test-query; 2) Auth: $\mathcal{A}$ correctly guesses credential pk_i is in the server and matches the user's password and executes the Test-query on the user's instance.

Game_0. This game corresponds to the real attack. Therefore, we have $\mathsf{Adv}^{ake}(\mathcal{A}) = |2Pr[Succ_0] - 1|$.

Game_1. All hash functions H_i are modeled as independent random oracles in Game_0. In this game, we implement these oracles by lazy sampling: each previously unseen input receives a uniformly sampled output of the prescribed length, while repeated queries receive the previously recorded output. As a result, we have

$$|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_0]| = 0.$$

Game_2. This game excludes the collisions on the choices of random values and hash oracle outputs. Each Execute-query generates at most two honest pseudonyms, and each Send-query generates at most one. Thus, at most $2q_{exe}+q_{send}$ honest pseudonyms are sampled from $\{0,1\}^{\ell}$. Let q_h bound the total number of distinct input–function pairs evaluated under the hash functions by both $\mathcal{A}$ and the honest algorithms (i.e., registration and offline phase). Let κ_H be the minimum output length of these functions. By the birthday bound, we have

$$|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_1]| \le \frac{(2q_{exe}+q_{send})^2}{2^{\ell+1}} + \frac{q_h^2}{2^{\kappa_H+1}}.$$

Game_3. This game is identical to Game_2, except that executions in which Auth occurs are assigned an independent fair success bit. Here, Auth concerns successful online username–password guessing, rather than arbitrary recovery of a registered credential. All protocol computations and query responses are otherwise unchanged. Consequently, we have $|\Pr[\mathsf{Succ}_3] - \Pr[\mathsf{Succ}_2]| \le \Pr_{\mathsf{Game}_2}[\mathsf{Auth}]$. Note that, $Pr(\mathsf{Auth})$ is constrained by the probability of online password guessing. Thus, we have

$$|\Pr[\mathsf{Succ}_3] - \Pr[\mathsf{Succ}_2]| \le C' \cdot q_{send}^{s'}.$$

Game_4. We replace the VOPRF component by its ideal execution and the simulator provided by the assumed component security definition. The public input remains $\mathcal{I} = \perp$. Under a fixed honest server-key context, ideal outputs are those of one consistent random function with output space $\{0,1\}^{\lambda}$. The same input under that context receives the same output during login. Credential key pairs are still obtained using $\mathsf{EncKG}(1^{\lambda}, Y)$. We employ an OPRF distinguisher $\mathcal{B}_O$ that runs $\mathcal{A}$ and uses its real-or-ideal component interface to simulate these invocations. It simulates the remaining qaPAKE operations, samples the Test bit independently, and outputs the success indicator of the resulting experiment. The assumed component theorem must cover this surrounding interaction and its guess accounting. Its real interface gives Game_3, and its ideal interface gives Game_4. Therefore, we have

$$|\Pr[\mathsf{Succ}_4] - \Pr[\mathsf{Succ}_3]| \le Adv_{\mathcal{A}}^{OPRF}(\lambda).$$

Game_5. In this game, $\mathcal{A}$ can obtain l_i and c_i. Due to the obliviousness of VPOPRF, even if $\mathcal{A}$ can obtain I_u and matching pw_u, $\mathcal{A}$ cannot predict Y_u. $\mathcal{A}$ can only obtain the l_i from the random oracle. If $\mathcal{A}$ can distinguish the ciphertext from the c_i returned by the simulator, $\mathcal{A}$ can disrupt the security of Enc. Hence, we have

$$|\Pr[\mathsf{Succ}_5] - \Pr[\mathsf{Succ}_4]| \le Adv_{\mathcal{A}}^{Enc}(\lambda).$$

Game_6. For each verification key pk_s, the challenger records the messages signed under that key in $\mathcal{M}_{pk_s}$. This game additionally rejects a user's final response if signature verification

succeeds for $m = pid_u \| ek_u \| ct_s$ but $m \notin \mathcal{M}_{pk_s}$. Denote this event in the preceding game by Forge. A replay of a signature on an already signed message is not included in this event. Therefore, we have

$$|\Pr[\mathsf{Succ}_6] - \Pr[\mathsf{Succ}_5]| \leq \Pr_{\mathsf{Game}_5}[\mathsf{Forge}].$$

Construct an EUF-CMA forger $\mathcal{B}_\Sigma$ that selects one of the at most N_B batch key-generation positions uniformly and places its challenge verification key at that position. All other signing key pairs are generated normally. The target batch ciphertexts and any prescribed consistency proof can be produced using the challenge verification key, credential material, and encryption randomness; the signing secret is not needed for these computations. Required signatures under the target key are obtained from the signing oracle, and the submitted messages are recorded. If Forge occurs under the selected key, $\mathcal{B}_\Sigma$ outputs the corresponding (m, σ_s). This is a valid signature on a message not submitted to its signing oracle. Since target signing secrets are not disclosed in the present experiment, we have

$$|\Pr[\mathsf{Succ}_6] - \Pr[\mathsf{Succ}_5]| \leq N_B Adv^{\mathsf{SIG}}_{\mathcal{A}}(\lambda).$$

Game_7. This game follows Game_6, except that it aborts if an honest server accepts an authentication request that was not emitted by an honest user and $\mathcal{A}$ has not queried H_4 on the corresponding input $pid_s \| pk_s \| ek_u$. The matching honest-request records exclude previously disclosed authentication values. Under the collision exclusions of Game_2, each server instance checks a fresh verification context at most once. Thus, a successful request of this kind requires guessing an unexposed output of H_4, consisting of $|\tau_u|$ uniformly random bits. Therefore, we have

$$|\Pr[\mathsf{Succ}_7] - \Pr[\mathsf{Succ}_6]| \leq q_{send}/2^{|\tau_u|}.$$

Game_8. We replace the KEM shared secret of each such honest source exchange by an independent $\widehat{z} \leftarrow \{0,1\}^{\kappa_z}$. The corresponding server and user computations use the same replacement. In particular, both the session key and the confirmation tag are computed using it: $SK = H_6(pid \| ct_s \| \widehat{z} \| pk_s \| ek_u)$ and $\tau_s = H_7(pid \| ct_s \| \widehat{z} \| pk_s \| ek_u)$. First, for matched honest ciphertexts, we use the recorded server shared secret in place of the honest user's decapsulation result. This changes an execution only on a KEM correctness failure. We employ at most N_U hybrids, indexed by honest user KEM key-generation positions. In hybrid j, the corresponding shared secrets for the first j positions are replaced, and the remaining ones are real.

For two consecutive hybrids, a KEM distinguisher $\mathcal{B}_j$ embeds its challenge encapsulation key $ek^\star$ at position j and generates all other key pairs itself. When the unique corresponding honest server encapsulation is needed, it obtains $(ct^\star, Z^\star)$ from its KEM challenge and uses these values in the server response. No unknown decapsulation result is needed for the selected user instance. A final response under a wrong-origin verification key is rejected by Game_5. A verified signature on an unrecorded message is rejected by Game_6. Any response that reaches decapsulation therefore contains $ct^\star$ from the recorded matching encapsulation, and $\mathcal{B}_j$ uses $Z^\star$.

If no matching server encapsulation occurs, no final response for this instance reaches decapsulation. A real KEM challenge gives the preceding hybrid, and an independent random challenge gives the next one. Summing over the positions, we have

$$|\Pr[\mathsf{Succ}_8] - \Pr[\mathsf{Succ}_7]| \leq N_U Adv^{\mathsf{KEM}}_{\mathcal{A}}(\lambda).$$

Game_9. This Game is same as Game_8 unless $\mathcal{A}$ queries H_6 or H_7 on some W_j. Let Ask denote the more general event that a correctly encoded query to either function contains some $\widehat{z}_j$ in its shared-secret field. In Game_9, the sampled keys and confirmation tags do not depend on the values $\widehat{z}_j$. These secrets therefore remain uniform and independent of the adversary's view and query list. The adversary makes at most q_h random-oracle queries in total, so the number of its queries to H_6 and H_7 combined is also at most q_h. There are at most N_U replacement secrets, each of length κ_z bits. Therefore, we have

$$|\Pr[\mathsf{Succ}_9] - \Pr[\mathsf{Succ}_8]| \leq N_U q_h / 2^{\kappa_z}.$$

For a fresh tested instance covered by these replacements, the session key is uniform and independent of the confirmation tag. Freshness excludes its disclosure through the tested instance or a matching instance, including one that accepts later. Thus, the real and random Test responses are identically distributed, and $\Pr[\mathsf{Succ}_9] = 1/2$. In all, according to Game_0 to Game_9, we have

$$\begin{aligned} Adv^{ake}_{\mathcal{A}}(\lambda) \leq\ & 2C' q^{s'}_{send} + 2Adv^{OPRF}_{\mathcal{A}}(\lambda) + 2Adv^{Enc}_{\mathcal{A}}(\lambda) \\ & + 2N_B Adv^{\mathsf{SIG}}_{\mathcal{A}}(\lambda) + 2N_U Adv^{\mathsf{KEM}}_{\mathcal{A}}(\lambda) + \frac{q_h^2}{2^{\kappa_H}} \\ & + \frac{(2q_{exe} + q_{send})^2}{2^\ell} + \frac{2q_{send}}{2^{|\tau_u|}} + \frac{2N_U q_h}{2^{\kappa_z}}. \end{aligned}$$

Theorem 2 (User anonymity)**.** *Let $Adv^{ua}_{\mathcal{A}}$ be the advantage of $\mathcal{A}$ breaking user anonymity. We have*

$$Adv^{ua}_{\mathcal{A}}(\lambda) \leq 2q_{exe} \cdot Adv^{OPRF}_{\mathcal{A}}(\lambda).$$

Proof: We employ similar notations and conventions to Theorem 1 to conduct a sequence of games, which starts from the real case Game_0 and ending at the ideal case Game_2.
Game_0. This game corresponds to the real attack. We have

$$\mathsf{Adv}^{ua}(\mathcal{A}) = |2Pr[Succ_0] - 1|.$$

Game_1. In this game, the VOPRF executions involving the challenge user are replaced by simulated executions guaranteed by VOPRF obliviousness. Suppose that $\mathcal{A}$ distinguishes Game_1 from Game_0. We construct an OPRF adversary that embeds its obliviousness challenge into one challenge-user VOPRF execution and simulates the remaining protocol operations honestly. A real challenge transcript corresponds to Game_0, whereas a simulated transcript corresponds to Game_1. Since at most q_{exe} challenge executions are involved, we have

$$|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_0]| \leq q_{exe} \cdot Adv^{OPRF}_{\mathcal{A}}(\lambda).$$

Game_2. We now consider the protocol transcript outside the VOPRF execution. Since both cre_0 and cre_1 are valid registered credentials, the challenge user finds a matching label in L in either challenge world. The matched positions

may be different, but the position is determined locally and is never sent to the server. For every authentication file in an honest broadcast batch, the corresponding ciphertext encrypts the same batch-specific verification key pk_s. Hence, regardless of whether cre_0 or cre_1 is used, the challenge user recovers the same pk_s. The remaining values generated by the challenge user are session-specific: RID_u is sampled freshly, $(dk_u, ek_u) \leftarrow \mathsf{KEM.Gen}(1^\lambda)$ is generated independently of the credential, and $pid_u = H_3(RID_u \| RID_s)$. Consequently, $\tau_u = H_4(pid_u \| pk_s \| ek_u)$ has the same distribution.

The subsequent server response depends on (pid_u, ek_u), the same batch-specific pk_s, and fresh session randomness, but not on the position of the challenge user's authentication file. Thus, after the VOPRF transcript has been simulated in Game_1, the complete protocol view of $\mathcal{A}$ is independent of b. Therefore,

$$|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_1]| = 0 \quad \text{and} \quad \Pr[\mathsf{Succ}_2] = \frac{1}{2}.$$

In all, according to Game_0 to Game_2 we have

$$Adv^{ua}_{\mathcal{A}}(\lambda) \leq 2q_{exe} \cdot Adv^{OPRF}_{\mathcal{A}}(\lambda).$$

Theorem 3 (Server anonymity)**.** *Let $Adv^{sa}_{\mathcal{A}}$ be the advantage of $\mathcal{A}$ breaking server anonymity. We have*

$$Adv^{sa}_{\mathcal{A}}(\lambda) \leq 2n_s Adv^{OPRF}_{\mathcal{A}}(\lambda) + 2n_s q_{exe} Adv^{\mathsf{SEnc}}_{\mathcal{A}}(\lambda).$$

Proof: Let $\mathcal{T}_0$ and $\mathcal{T}_1$ be the two credential sets of the server, where $|\mathcal{T}_0| = |\mathcal{T}_1| = n_s$. We employ similar notations and conventions to Theorem 2 to conduct a sequence of games. Game_0. This game corresponds to the real attack. We have

$$\mathsf{Adv}^{sa}(\mathcal{A}) = |2Pr[Succ_0] - 1|.$$

Game_1. We first replace the VOPRF-derived credential material associated with entries on which $\mathcal{T}_0$ and $\mathcal{T}_1$ differ. For an authentication file corresponding to input p_i, the user-side credential is derived from $Y_i = F_K(p_i)$ as $(sk_i, pk_i) \leftarrow \mathsf{EncKG}(1^\lambda, Y_i)$, and its broadcast label is $l_i = H_2(pk_i)$. For each differing database position, we replace the corresponding VOPRF output by the output provided by the ideal VOPRF experiment and derive pk_i and l_i from this value in the same way. Suppose that $\mathcal{A}$ distinguishes two consecutive hybrids. We construct an OPRF adversary that embeds its real-or-ideal challenge at the changed position and simulates the remaining protocol honestly. A real OPRF output gives the preceding hybrid, whereas the ideal output gives the next one. Since the two credential sets have equal size, at most n_s such replacements are required. Therefore, we have

$$|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_0]| \leq n_s Adv^{OPRF}_{\mathcal{A}}(\lambda).$$

Game_2. This game further replaces the broadcast ciphertexts associated with the differing entries. The ciphertext corresponding to an authentication file is $c_i = \mathsf{SEnc}(H_1(pk_i), pk_s)$. For entries common to $\mathcal{T}_0$ and $\mathcal{T}_1$, including any credential used in an allowed Execute-query, the ciphertexts are left unchanged. For each remaining differing entry, we replace the encryption of pk_s by an encryption of a fixed equal-length dummy message under the same derived key. No user involved in an allowed challenge-server execution selects such an entry, because Execute-queries with distinguishing users are excluded by the server-anonymity experiment.

If $\mathcal{A}$ distinguishes two consecutive ciphertext hybrids, $\mathcal{A}$ uses the corresponding ciphertext as its challenge and simulates the remaining entries honestly. Thus, each replacement is bounded by $Adv^{\mathsf{SEnc}}_{\mathcal{A}}(\lambda)$. There are at most n_s replaced ciphertexts in each observed broadcast and at most q_{exe} Execute-queries, and we have

$$|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_1]| \leq n_s q_{exe} Adv^{\mathsf{SEnc}}_{\mathcal{A}}(\lambda).$$

Game_3. After the preceding replacements, the entries on which $\mathcal{T}_0$ and $\mathcal{T}_1$ differ consist of challenge-independent credential labels and ciphertexts. The entries common to both sets are identical in the two challenge worlds, and the two databases have the same size. Moreover, every user allowed to execute with the challenge server has the same membership status in $\mathcal{T}_0$ and $\mathcal{T}_1$. Hence, its local matching result and authentication outcome are also independent of b. The remaining online values, including RID_s, the batch-specific (sk_s, pk_s), the KEM encapsulation, the signature, and the session confirmation value, are generated independently of which challenge database was selected. Thus, the view of $\mathcal{A}$ is identically distributed for $b = 0$ and $b = 1$, and $\Pr[\mathsf{Succ}_3] = \frac{1}{2}$.

In all, according to Game_0 to Game_3, we have

$$Adv^{sa}_{\mathcal{A}}(\lambda) \leq 2n_s Adv^{OPRF}_{\mathcal{A}}(\lambda) + 2n_s q_{exe} Adv^{\mathsf{SEnc}}_{\mathcal{A}}(\lambda).$$

TABLE IV: Security level of our qaPAKE

Parameter	n	q	β	Hermite factor	Security level
Params I	512	268,435,399	342	1.0044	92-bits
Params II	595	268,435,399	478	1.0035	128-bits

† n be the dimension; q be the original modulus; β be the BKZ block size.

TABLE V: Experimental settings for evaluating our qaPAKE.

	Client	Server
Device	Huawei P30	Workstation
Processor	Kirin 980	Intel Core i5-14400F
Memory	8 GB	16 GB
Disk capacity	128 GB	1 TB
Operating syst.	HarmonyOS 4.0	Ubuntu 20.04.6 x64
Round-trip latency	WAN network delay $\approx 15ms$	
Energy measurement	Huawei P30, measured at 4 V and 489 mA	

A. Quantum security estimation of qaPAKE parameters

In qaPAKE, the VPOPRF is constructed from lattice-based primitives, while the session-specific key establishment and signature components are instantiated with ML-KEM and ML-DSA, respectively. The broadcast ciphertext set C is generated using symmetric encryption and is therefore not part of the lattice-parameter estimation below. The formal security proofs model the hash functions $H_0, \ldots, H_7$ as random oracles.

Let the lattice dimension $n = \lambda^c$, where $c > 2$ is a constant. Set the bit-length of $\mathbf{x}$ to $L = \lambda$, the secret and error distribution $\sigma = \mathsf{poly}(n)$, and $\sigma' = \sigma^2 n^2 \cdot \lambda^{\omega(1)}$. Let the original modulus $q = p \cdot \prod_{i=1}^{n'} p_i$, where the rounding modulus $p_i = \sigma' \cdot \omega\left(\sqrt{nn' \log q \log n'}\right)$. There is $q = p \cdot \sigma' \cdot \lambda^{\omega(1)}$ [44]. We set $q = 2^{28} - 57$, $n = 595$, and $\sigma = \alpha \cdot q/\sqrt{2\pi}$. We employ the "lwe-estimator" scripts[1] to evaluate the security level of

[1]The "lwe-estimator" is available at https://bitbucket.org/malb/lweestimator.

TABLE VI: A comparison of password-only aPAKE protocols in terms of computation and communication overheads.†

Protocol	Technology	Computation overhead			Communication overhead
		User	Server (online)	Server (offline)	
Vasco et al. [9]‡	OPRF+KE	$T_O + 4T_H + 2T_E$	$T_O + 2T_H + T_E$	$NT_P + 2NT_H + NT_E$	$N\|H\| + (N+1)\|E\|$
Viet et al. [17] ‡	Oblivious transfer	$4T_H + 6T_{exe}$	$(N+3)T_H + (3N+1)T_{exe}$ ‡	$(N+1)T_{exe}$	$(N+2)\|G\| + (N+1)\|H\|$
Yang and Zhang* [18]	Key exchange	$2T_H + 4T_{exe}$	$2T_H + 3T_{exe}$	NT_{exe}	$(N+3)\|G\| + \|H\|$
Shin et al. [19]‡	Encryption	$3T_H + T_E + 3T_{exe}$	$3T_H + 4T_{exe}$	$(N+1)T_H$+NT_E+$(N+1)T_{exe}$	$3\|G\| + 2\|H\| + N\|E\|$
Our qaPAKE	OPRF+KE	$T_O + 8T_H + T_E + T_K$	$T_O + 5T_H + T_K + T_S$	$NT_H + NT_E$	$(N+4)\|H\| + N\|E\|$

† KE=key exchange; OPRF=oblivious pseudorandom function. Let T_H, T_P, T_O, T_E, and T_{exe} denote the time costs of hash evaluation, PRF evaluation, OPRF computation, symmetric encryption/decryption, and exponentiation, respectively. T_K denotes the cost of key encapsulation mechanism; and T_S denotes the cost of signature. Let $|G|$, $|H|$, and $|E|$ denote the bit lengths of an encoded group element, a hash output, and an encryption ciphertext, respectively.
‡ The server stores passwords in plaintext. $\mathcal{A}$ corrupting the server directly obtains users' passwords without performing an offline password-guessing attack.
∗ Yang and Zhang's protocol [18] is device-free but not strictly password-only, as users must remember a credential index in addition to their passwords.

TABLE VII: Mobile performance of qaPAKE for $N = 100$ under the honest-but-curious and malicious settings.†

Metric	**Honest-but-curious**			**Malicious**		
	Client	**Server offline**	**Server online**	**Client**	**Server offline**	**Server online**
Runtime (ms)	65.58	0.32	8.07	401.21	856.72	17.44
Memory usage (MB)	84.52	82.11	11.68	85.57	83.16	12.73
Energy consumption (mJ)	128.31	20.81	2,451.54	784.81	5,798.65	9,507.53
Bandwidth	$U \to S$	$S \to U$	**Total**	$U \to S$	$S \to U$	**Total**
Communication overhead (KiB)	68.35	272.16	340.51	77.49	1357.58	1435.07

† $U \to S$ and $S \to U$ denote the communication from the user to the server and from the server to the user, respectively. We measure runtime and memory usage for each phase on the client and server settings in Table V. Communication overhead includes all traffic exchanged during authentication.

our protocol. In particular, we follow the estimator's noise-rate definition $\alpha = \sigma \cdot \sqrt{2\pi}/q$, and the OPRF parameter settings [44], [39]. To obtain more conservative parameters, we adopt the core-SVP methodology using the classical cost $2^{0.292*\beta}$ and quantum cost $2^{0.268*\beta}$, and determine the parameters to meet the target estimated quantum security level. As shown in Table IV, qaPAKE can provide 92-bit quantum security with Param I, and 128-bit quantum security with Param II.

The lattice-parameter estimates above are separate from our security proofs, which are conducted in the classical random oracle model. Our analysis does not cover quantum superposition queries to the hash oracles. The security proof in the quantum random oracle model remains future work.

Further discussion. Our formal security analysis is conducted in the random oracle model (ROM) [94], where the hash functions used in the protocol are modeled as ideal random functions. The use of the ROM does not change the quantum hardness of the underlying lattice assumptions; ROM and QROM mainly differ in the adversary's access model to the hash oracle. In particular, the lattice assumptions underlying the VPOPRF [44], [42], ML-KEM [73], and ML-DSA [74] remain quantum-resistant independently of whether the hash functions are analyzed in the ROM or QROM.

Boneh et al. [95] introduced the quantum random oracle model (QROM), which further allows a quantum adversary to query the modeled hash function in quantum superposition. Thus, QROM captures a stronger hash-oracle access capability than the classical ROM. Grover's algorithm [89] provides a quadratic speedup for generic unstructured search, including preimage search, which can be accommodated in practice by selecting appropriate hash-output lengths.

The ROM has been widely adopted for the formal analysis of lattice-based authentication and key-establishment protocols [44], [84]. In practical instantiations, the random oracles in qaPAKE can be instantiated using SHAKE-based hash functions [37], while the algebraic components are instantiated with quantum-resistant lattice primitives. As the first step towards a quantum-resistant password-only anonymous authenticated key exchange, our work follows the ROM to provide security proof for qaPAKE. In addition, we will focus on exploring the security under QROM in future work.

VI. Performance Evaluation

In this section, we evaluate the efficiency of our proposed qaPAKE and compare it with four representative classes of state-of-the-art password-only aPAKE protocols [9], [17], [18], [19]. Our evaluation covers computation overhead, communication cost, and usability, with a particular focus on the device-free property enabled by password-only authentication. We evaluate whether the design choices of qaPAKE remain practical on resource-constrained mobile devices. Our experiments focus on three questions. First, we examine whether the online protocol can be executed efficiently on a commodity smartphone. Second, we quantify the additional cost of protecting against a malicious server. Finally, we study how computation, memory, and communication scale with the credential database size.

Overhead analysis. We first conduct an operation-count-based overhead analysis of qaPAKE and compare it with representative password-only aPAKE protocols [17], [18], [19], [9]. For a consistent comparison, all protocols are evaluated under the honest-but-curious model, excluding all zero-knowledge proof costs. Table VI shows the computation costs in terms of standard cryptographic operations, together with the communication overheads. Compared with Viet et al. [17], whose online server computation grows linearly with N, qaPAKE incurs cryptographic computation costs independent of the credential-set size N at both the user and the online server.

Next, we empirically evaluate the practical performance of qaPAKE. We first measure its computation and communication costs on a real mobile setting under both the honest-but-curious and malicious settings. We then examine its scalability

TABLE VIII: Scalability of qaPAKE with the credential database size.†

Credential database size N	Computation overhead (ms)						Memory usage (MB)				Communication overhead (KiB)			
	Honest-but-curious			Malicious			Honest-but-curious		Malicious		Honest-but-curious		Malicious	
	Client	Server (offline)	Server (online)	Client	Server (offline)	Server (online)	Client	Server	Client	Server	$U \to S$	$S \to U$	$U \to S$	$S \to U$
10	65.59	0.12	8.07	401.22	89.21	17.44	84.45	82.01	84.57	82.13	68.35	151.58	77.49	278.63
20	65.58	0.15	8.07	401.20	174.50	17.44	84.49	82.00	84.72	82.22	68.35	164.98	77.49	398.52
50	65.58	0.19	8.07	401.21	430.31	17.44	84.55	82.05	85.08	82.58	68.35	205.17	77.49	758.17
100	65.58	0.32	8.07	401.21	856.72	17.44	84.52	82.11	85.57	83.16	68.35	272.16	77.49	1,357.58
1000	65.59	1.99	8.07	401.22	8531.53	17.44	85.74	83.67	96.15	94.08	68.35	1,478.02	77.49	12,147.03

† $U \to S$ and $S \to U$ denote user-to-server and server-to-user communication, respectively. All measurements employ the client and server settings specified in Table V. Computation overhead is reported separately for the client, the server offline phase, and the server online phase. Memory usage denotes the maximum phase-local resident-set size.

TABLE IX: Measured end-to-end latency and total communication overhead for various credential database sizes.†

N	End-to-end latency (ms)		TCP data packets		Total communication (KiB)	
	HBC	Malicious	HBC	Malicious	HBC	Malicious
10	129.18	485.33	155	251	219.93	356.12
20	130.26	495.13	164	335	233.33	476.01
50	133.56	524.61	192	587	273.52	835.66
100	139.04	573.71	239	1,008	340.51	1,435.07
1,000	237.84	1,457.59	1,085	8,575	1,546.37	12,224.52

† N = Credential database size; HBC = Honest-but-curious; TCP = Transmission control protocol. We employ a symmetric bandwidth of 100 Mbit/s and a round-trip time of 15 ms. Latency covers the online phase.

as the number of credentials in the server database increases. Finally, we implement representative classical password-only aPAKE schemes [9], [17] on the same experimental setting and provide a direct empirical comparison with our qaPAKE. As shown in Table V, we implement qaPAKE in C using NTL 11.5.1 and "liboqs" open-quantum-safe v0.15.0 library. The client runs on a Huawei P30 equipped with a Kirin 980, 8 GB RAM, and HarmonyOS 4.0 with Termux 0.118.3, whereas the server runs on an Intel Core i5-14400F @ 2.50 GHz with 16 GB RAM and Ubuntu 20.04.6 x64.

Mobile Performance. We first evaluate the performance of qaPAKE on a real mobile device. Table VII shows the runtime, memory footprint, communication overhead, and energy consumption for $N = 100$ under both the honest-but-curious and malicious settings, while the impact of increasing the credential database size is examined separately in Table VIII.

The honest-but-curious measurements exclude all zero-knowledge proofs. In the malicious setting, client computation includes generating the OPRF request proof and verifying the OPRF response proof and the batch-consistency proof. Server online computation includes verifying the OPRF request proof and generating the OPRF response proof, while server offline computation includes generating the batch-consistency proof. Communication overhead includes the transmitted proofs.

In the honest-but-curious setting, online computation takes 65.58 ms on the Huawei P30 and 8.07 ms on the server, with 0.32 ms of server-side batch precomputation. As in earlier aPAKE designs [9], [19], precomputation moves database-dependent preparation outside the login path. The client and server phase-local peak RSS values are 84.52 MB and 82.11 MB, respectively. Using the measured operating values of approximately 4 V and 489 mA, the estimated energy consumed during client computation is 128.3 mJ.

Communication constitutes the main online scalability tradeoff of the full-download architecture. At $N = 100$, the server and client transmit 272.16 KiB and 68.35 KiB, respectively, totaling 340.51 KiB, including initialization traffic. Similar to the OPRF-based approach of Vasco et al. [9], local credential selection avoids requiring the client to maintain a database index. In qaPAKE, this is enabled by downloading the vectors (L, C), whose size grows linearly with N. Thus, in qaPAKE, eliminating persistent client-side authentication state entails database-dependent server-to-client traffic.

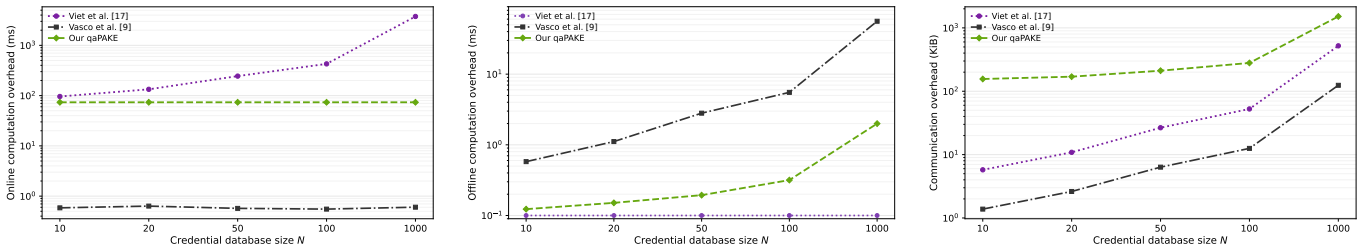
The malicious setting adds OPRF validity proofs and the batch-consistency proof π_{eq}, which establishes that all broadcast ciphertexts encrypt the same pk_s and prevents credential-dependent key tagging. At $N = 100$, client and server online computation increase to 401.21 ms and 17.44 ms, respectively, while server-side batch precomputation takes 856.72 ms. The client and server phase-local peak RSS values are 85.57 MB and 83.16 MB. Thus, malicious security increases the computational cost, although client computation remains below half a second on the Huawei P30 in this experiment.

$S \to U$ and $U \to S$ traffic increase to 1,357.58 KiB and 77.49 KiB, respectively, totaling 1,435.07 KiB. Of the additional 1,094.56 KiB per authentication, $S \to U$ transmission accounts for 1,085.42 KiB, or approximately 99.2%. The dominant additional communication cost therefore lies in downstream proof transmission, while database-dependent consistency-proof generation remains in the offline phase.

Scalability. In the full-download architecture, the broadcast size increases with the number N of registered credentials. We therefore evaluate how increasing N affects the online and offline computational costs, as well as the communication overheads, of both the user and the server. Table VIII shows the results for database sizes $N \in \{10, 20, 50, 100, 1,000\}$.

The results show that online computation remains nearly unchanged as N increases in both settings. At $N = 1,000$, client and server online computation take 65.59 ms and 8.07 ms in the honest-but-curious setting, and 401.22 ms and 17.44 ms in the malicious setting. Database growth instead increases server preprocessing, which reaches 1.99 ms and 8,531.53 ms, respectively. The offline–online separation therefore limits the effect of database growth on login computation, while malicious security imposes a larger batch-preparation cost.

Memory growth differs between the two settings. Across the tested range, the client and server memory usage increase by 1.29 MB and 1.66 MB in the honest-but-curious setting. In the malicious setting, the corresponding increases are



(a) Online computation overhead. (b) Offline computation overhead. (c) Communication overhead.

Fig. 5: Comparison of qaPAKE with the password-only aPAKE schemes [9], [17] under the experimental setting in Table V. (a) shows the online computation overhead; (b) compares offline computation overhead, with Viet et al. [17] plotted at zero; (c) shows the total communication overhead. The vertical axes use logarithmic scales to facilitate comparisons across different orders of magnitude.

11.58 MB and 11.95 MB, reaching 96.15 MB and 94.08 MB at $N = 1,000$. Thus, memory usage remains below 100 MB at both endpoints throughout the tested range, but the malicious-setting memory cost is more sensitive to database growth.

Communication remains the principal online scalability cost. Client-to-server traffic stays constant at 68.35 KiB and 77.49 KiB in the two settings, whereas server-to-client traffic grows linearly with N. At $N = 1000$, the latter reaches 1,478.02 KiB in the honest-but-curious setting and 12,147.03 KiB in the malicious setting. The batch-consistency proof introduces additional database-dependent traffic in the malicious setting. In qaPAKE, the client downloads (L, C) to select its credential locally, avoiding the need to store a database index or disclose a stable lookup identifier. The measured computation and memory costs support computational feasibility on the evaluated mobile setting for database sizes up to $N = 1000$, while downstream bandwidth and batch-preparation costs affect deployment. For modest user populations, qaPAKE could provide anonymous authentication for employee surveys [96] and Wi-Fi access [11].

Table IX shows the latency and TCP packet counts measured using our implementation on a Huawei P30 and a server. The tests focus on the online phase, with a bandwidth of 100 Mbit/s and a network delay of 15 ms. As the database size increases from 10 to 1,000 credentials, the latency rises from 129.18 ms to 237.84 ms under the honest-but-curious setting and from 485.33 ms to 1,457.59 ms under the malicious model. The corresponding packet counts increase from 155 to 1,085 and from 251 to 8,575, respectively. These increases reflect the growing cost of database transmission, particularly the additional proof transmission overhead under the malicious setting, while online computation time remains unchanged.

Fig. 5 compares our qaPAKE with the first password-only aPAKE of Viet et al. [17] and the OPRF-based scheme of Vasco et al. [9], using the settings in Table V under the honest-but-curious setting. Compared with Viet et al., our qaPAKE improves server-side online scalability: at $N = 1000$, its server online computation is 458 times faster, with only 6.53 ms increase in client computation. Compared with Vasco et al. [9], our qaPAKE provides quantum resistance and reduces server preprocessing time from 56.04 ms to 1.99 ms, but incurs higher online computation and communication costs. Its mobile computation nevertheless remains below 66 ms across all tested database sizes, supporting computational feasibility on the evaluated smartphone while exposing the bandwidth cost of the construction.

Function. Table I compares representative aPAKE protocols in terms of storage requirements, user inputs, cryptographic techniques, security models, round complexity, quantum resistance, and online costs per authentication. Among the compared protocols, existing password-only schemes do not provide quantum resistance, whereas the quantum-resistant schemes require persistent client-side secret storage.

Our qaPAKE combines password-only authentication with quantum resistance, requiring users to remember only their username and password. Compared with the OPRF-based scheme of Vasco et al. [9], qaPAKE reduces the authentication interaction from three rounds and six flows to two rounds and four flows. Its server-side online computation is independent of the credential database size, although the full-download architecture entails a server-to-user communication cost that grows linearly with the credential database size N.

A key usability advantage of password-only aPAKE is that users need not maintain persistent cryptographic secrets on their devices. Unlike password-protected aPAKEs [14], [15], [13] or constructions requiring user-side secret material [26], [25], [24], this approach allows users to authenticate across desktops, laptops, and mobile devices without transferring or synchronizing locally stored cryptographic credentials.

Our qaPAKE preserves this portability while providing quantum resistance, supporting password-only authentication across multiple devices without additional mechanisms for provisioning or recovering persistent client-side cryptographic secrets. This makes qaPAKE attractive for real-world deployments requiring both high usability and long-term security.

VII. CONCLUSION

This paper addresses the challenge of constructing a quantum-safe password-only aPAKE. We have proposed a lightweight VPOPRF to re-randomize passwords, effectively resisting offline password-guessing attacks. Building on this foundation, we, *for the first time*, have designed a quantum-safe password-only aPAKE over the Ring-LWE problem, named qaPAKE. Through rigorous theoretical analysis and experimental comparisons, we have demonstrated the security and effectiveness of our qaPAKE, achieving practical computational efficiency on mobile devices and well balancing the demanding requirements of quantum-resistance and password-only properties. As a result, our work guides researchers aiming to design new password-only aPAKE protocols in the post-quantum era.

REFERENCES

[1] J. Bonneau, C. Herley, P. Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *Proc. IEEE S&P 2012*, pp. 553–567.

[2] ——, "Passwords and the evolution of imperfect authentication," *Commun. ACM*, vol. 58, no. 7, pp. 78–87, 2015.

[3] S. Jarecki, H. Krawczyk, and J. Xu, "Opaque: an asymmetric pake protocol secure against pre-computation attacks," in *Proc. EUROCRYPT 2018*, pp. 456–486.

[4] Y. Lyu, S. Liu, and S. Han, "Efficient asymmetric pake compiler from kem and ae," in *Proc. ASIACRYPT 2025*, pp. 34–65.

[5] M. Bellovin and M. Merritt, "Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise," in *Proc. ACM CCS 1993*, pp. 244–250.

[6] V. Shoup, "Security analysis of itspake2+," in *Proc. TCC 2020*, pp. 31–60.

[7] Y. Lyu, S. Liu, and S. Han, "Universal composable password authenticated key exchange for the post-quantum world," in *Proc. EUROCRYPT 2024*, pp. 120–150.

[8] Y. Gu, S. Jarecki, and H. Krawczyk, "KHAPE: Asymmetric PAKE from key-hiding key exchange," in *Proc. CRYPTO 2021*, pp. 701–730.

[9] M. Vasco, A. del Pozo, and C. Soriente, "A key for john doe: Modeling and designing anonymous password-authenticated key exchange protocols," *IEEE Trans. Depend. Secur. Comput.*, vol. 18, no. 3, pp. 1336–1353, 2019.

[10] H. Lian, Y. Yang, and Y. Zhao, "Efficient and strong symmetric password authenticated key exchange with identity privacy for iot," *IEEE Internet Things J.*, vol. 10, no. 6, pp. 4725–4734, 2022.

[11] A. Cassola, E. Blass, and G. Noubir, "Authenticating privately over public wi-fi hotspots," in *Proc. ACM CCS 2015*, pp. 1346–1357.

[12] "ISO/IEC 20009-4:anonymous entity authentication part 4: Mechanisms based on weak secrets," in *ISO/IEC JTC 1/SC 27, Tech. Rep., 2017*.

[13] Z. Zhang, K. Yang, X. Hu, and Y. Wang, "Practical anonymous password authentication and tls with anonymous client authentication," in *Proc. ACM CCS 2016*, pp. 1179–1191.

[14] Y. Yang, J. Zhou, J. Weng, and F. Bao, "A new approach for anonymous password authentication," in *Proc. ACSAC 2009*, pp. 199–208.

[15] Y. Yang, J. Zhou, J. Wong, W. Jun, and F. Bao, "Towards practical anonymous password authentication," in *Proc. ACSAC 2010*, pp. 59–68.

[16] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *Proc. SCN 2003*, pp. 268–289.

[17] D. Viet, A. Yamamura, and H. Tanaka, "Anonymous password-based authenticated key exchange," in *Proc. INDOCRYPT 2005*, pp. 244–257.

[18] J. Yang and Z. Zhang, "A new anonymous password-based authenticated key exchange protocol," in *Proc. INDOCRYPT 2008*, pp. 200–212.

[19] S. Shin, K. Kobara, and H. Imai, "Very-efficient anonymous password-authenticated key exchange and its extensions," in *Proc. AAECC 2009*, pp. 149–158.

[20] X. Hu, J. Zhang, Z. Zhang, and J. Xu, "Universally composable anonymous password-authenticated key exchange," *Sci. China Inf. Sci.*, vol. 60, no. 5, pp. 52 107–52 119, 2017.

[21] X. Hu, J. Zhang, Z. Zhang, and F. Liu, "Anonymous password-authenticated key exchange protocol in the standard model," *Wirel. Pers. Commun.*, vol. 96, pp. 1451–1474, 2017.

[22] A. Shahidinejad, J. Abawajy, and S. Huda, "Quantum-proof anonymous key exchange with perfect forward secrecy for mobile devices," *IEEE Trans. Consum. Electron.*, vol. 71, no. 3, pp. 8425–8432, 2025.

[23] R. Mani, D. Chaudhary, A. Padmavathi, and C.-C. Lee, "Security analysis and designing unilateral and bilateral two party ideal-lattice based authenticated key establishment protocol for anonymous mobile communication," *Comput. Electr. Eng.*, vol. 128, pp. 1–10, 2025.

[24] R. Ding, C. Cheng, and Y. Qin, "Further analysis and improvements of a lattice-based anonymous pake scheme," *IEEE Syst. J.*, vol. 16, no. 3, pp. 5035–5043, 2022.

[25] V. Dabra, A. Bala, and S. Kumari, "LBA-PAKE: Lattice-based anonymous password-authenticated key exchange for mobile devices," *IEEE Syst. J.*, vol. 15, no. 4, pp. 5067–5077, 2020.

[26] Q. Feng, D. He, S. Zeadally, N. Kumar, and K. Liang, "Ideal lattice-based anonymous authentication protocol for mobile devices," *IEEE Syst. J.*, vol. 13, no. 3, pp. 2775–2785, 2018.

[27] Z. Liu, Y. Zhao, L. Kang, Z. Zhang, Y. Wang, and L. Zhu, "Highly reliable anonymous cloud synchronization," *IEEE Trans. Inf. Forensics Security*, 2026.

[28] S. Shin, K. Kobara, and H. Imai, "A secure threshold anonymous password-authenticated key exchange protocol," in *Proc. IWSEC 2007*, pp. 444–458.

[29] W. Tzeng, "Efficient 1-out-n oblivious transfer schemes," in *Proc. PKC 2002*, pp. 159–171.

[30] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, pp. 79–99, 2018.

[31] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proc. FOCS 1994*, pp. 124–134.

[32] A. Banerjee, T. Reddy.K, D. Schoinianakis, T. Hollebeek, and M. Ounsworth, "Post-quantum cryptography for engineers," https://datatracker.ietf.org/doc/draft-ietf-pquip-pqc-engineers/, 2024.

[33] NIST, *Post-Quantum Cryptography: Selected Algorithms*, March 2025, https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms.

[34] D. Moody, R. Perlner, A. Regenscheid, A. Robinson, and D. Cooper, "Transition to post-quantum cryptography standards," https://nvlpubs.nist.gov/nistpubs/ir/2024/NIST.IR.8547.ipd.pdf, 2024.

[35] G. Alagic *et al.*, "Status report on the fourth round of the nist post-quantum cryptography standardization process," https://csrc.nist.gov/projects/post-quantum-cryptography, 2024.

[36] Z. Li, D. Wang, and E. Morais, "Quantum-safe round-optimal password authentication for mobile devices," *IEEE Trans. Depend. Secur. Comput.*, vol. 19, no. 3, pp. 1885–1899, 2020.

[37] E. Alkim, L. Ducas, T. Poppelmann, and P. Schwabe, "Post-quantum key exchange a new hope," in *Proc. USENIX Sec. 2016*, pp. 327–343.

[38] J. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the tls protocol from the ring learning with errors problem," in *Proc. IEEE S&P 2015*, pp. 553–570.

[39] M. F. Esgin, R. Steinfeld, E. Tairi, and J. Xu, "LeOPaRd: Towards practical post-quantum oblivious PRFs via 2HashDH paradigm," in *Proc. CRYPTO 2026*, pp. 35–67.

[40] N. Alamati, G.-V. Policharla, S. Raghuraman, and P. Rindal, "Improved alternating-moduli prfs and post-quantum signatures," in *Proc. CRYPTO 2024*, pp. 274–308.

[41] L. Heimberger, T. Hennerbichler, F. Meisingseth, S. Ramacher, and C. Rechberger, "Oprfs from isogenies: designs and analysis," in *Proc. AsiaCCS 2024*, pp. 575–588.

[42] M. R. Albrecht, A. Davidson, A. Deo, and D. Gardham, "Crypto dark matter on the torus: Oblivious prfs from shallow prfs and tfhe," in *Proc. EUROCRYPT 2024*, pp. 447–476.

[43] I. Dinur, S. Goldfeder, T. Halevi, Y. Ishai, M. Kelkar, V. Sharma, and G. Zaverucha, "MPC-friendly symmetric cryptography from alternating moduli: candidates, protocols, and applications," in *Proc. CRYPTO 2021*, pp. 517–547.

[44] M. R. Albrecht, A. Davidson, A. Deo, and N. P. Smart, "Round-optimal verifiable oblivious pseudorandom functions from ideal lattices," in *Proc. PKC 2021*, pp. 261–289.

[45] D. Boneh, D. Kogan, and K. Woo, "Oblivious pseudorandom functions from isogenies," in *Proc. ASIACRYPT 2020*, pp. 520–550.

[46] N. Tyagi, S. Celi, T. Ristenpart, N. Sullivan, S. Tessaro, and C. A. Wood, "A fast and simple partially oblivious prf, with applications," in *Proc. EUROCRYPT 2022*, pp. 674–705.

[47] C. Hazay, "Oblivious polynomial evaluation and secure set-intersection from algebraic prfs," *J. Cryptol.*, vol. 31, no. 2, pp. 537–586, 2018.

[48] J. Camenisch and A. Lehmann, "Privacy-preserving user-auditable pseudonym systems," in *Proc. EuroS&P 2017*, pp. 269–284.

[49] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu, "Toppss: cost-minimal password-protected secret sharing based on threshold oprf," in *Proc. ACNS 2017*, pp. 39–58.

[50] A. Everspaugh, R. Chaterjee, S. Scott, A. Juels, and T. Ristenpart, "The pythia {PRF} service," in *Proc. USENIX Sec. 2015*, pp. 547–562.

[51] S. Jarecki, A. Kiayias, and H. Krawczyk, "Round-optimal password-protected secret sharing and t-pake in the password-only model," in *Proc. ASIACRYPT 2014*, pp. 233–253.

[52] S. Jarecki and X. Liu, "Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection," in *Proc. TCC 2009*, pp. 577–594.

[53] C. Hazay and Y. Lindell, "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries," in *Proc. TCC 2008*, pp. 155–175.

[54] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold, "Keyword search and oblivious pseudorandom functions," in *Proc. TCC 2005*, pp. 303–324.

[55] D. Wang, Y. Zou, Q. Dong, Y. Song, and X. Huang, "How to attack and generate honeywords," in *Proc. IEEE S&P 2022*, pp. 966–983.

[56] D. Goodin, "Meta pays the price for storing hundreds of millions of passwords in plaintext," https://arstechnica.com/security/2024/09/meta-slapped-with-101-million-fine-for-storing-passwords-in-plaintext, 2024.

[57] V. Petkauskas, "Rockyou2024: 10 billion passwords leaked in the largest compilation of all time," https://cybernews.com/security/rockyou2024-largest-password-compilation-leak, 2024.
[58] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: An underestimated threat," in *Proc. ACM CCS 2016*, pp. 1242–1254.
[59] J. Jiang and D. Wang, "Qpase: Quantum-resistant password-authenticated searchable encryption for cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 4231–4246, 2024.
[60] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proc. CRYPTO 2004*, pp. 41–55.
[61] M. Chase, S. Meiklejohn, and G. Zaverucha, "Algebraic macs and keyed-verification anonymous credentials," in *Proc. ACM CCS 2014*, pp. 1205–1216.
[62] J. Ding, X. Xie, and X. Lin, "A simple provably secure key exchange scheme based on the learning with errors problem," Cryptology ePrint Archive, Paper 2012/688, 2012. [Online]. Available: https://eprint.iacr.org/2012/688
[63] J. Ding, S. Alsayigh, R. Saraswathy, S. Fluhrer, and X. Lin, "Leakage of signal function with reused keys in rlwe key exchange," in *Proc. ICC 2017*, pp. 1–6.
[64] D. Dharminder and K. P. Chandran, "LWESM: learning with error based secure communication in mobile devices using fuzzy extractor," *J. Ambient Intell. Humaniz. Comput.*, vol. 11, no. 10, pp. 4089–4100, 2020.
[65] U. Kumar, M. Garg, S. Kumari, and D. Dharminder, "A construction of post quantum secure and signal leakage resistant authenticated key agreement protocol for mobile communication," *Trans. Emerg. Telecommun. Technol.*, vol. 34, no. 1, pp. 46–60, 2023.
[66] S. Banerjee, M. Nikhilesh, A. Sharma, A. G. Sreedevi, S. Rana, and D. Chaudhary, "Quantum safe construction of authentication and key exchange based on module lattices," in *Proc. IEEE GCAT 2023*, pp. 1–6.
[67] D. Chaudhary and C.-C. Lee, "Anonymous quantum-safe secure and authorized communication protocol under dynamic identities for Internet of Drones," *Comput. Electr. Eng.*, vol. 120, pp. 1–14, 2024.
[68] D. Dharminder, P. K. Dadsena, P. Gupta, and S. Sankaran, "A post quantum secure construction of an authentication protocol for satellite communication," *Int. J. Satell. Commun. Netw.*, vol. 41, no. 1, pp. 14–28, 2023.
[69] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 11, pp. 2776–2791, 2017.
[70] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *J. ACM*, vol. 56, no. 6, pp. 1–40, 2009.
[71] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proc. ACM STOC 2008*, pp. 197–206.
[72] J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs, "Learning with rounding, revisited: New reduction, properties and applications," in *Proc. CRYPTO 2013*, pp. 57–74.
[73] NIST, *Module-Lattice-Based Key-Encapsulation Mechanism Standard*, August 2024, https://csrc.nist.gov/pubs/fips/203/final.
[74] ——, *Module-Lattice-Based Digital Signature Standard*, August 2024, https://csrc.nist.gov/pubs/fips/204/final.
[75] V. Lyubashevsky, N. K. Nguyen, and M. Plançon, "Lattice-based zero-knowledge proofs and applications: shorter, simpler, and more general," in *Proc. CRYPTO 2022*, pp. 71–101.
[76] C. Peikert and S. Shiehian, "Noninteractive zero knowledge for np from (plain) learning with errors," in *Proc. CRYPTO 2019*, pp. 89–114.
[77] A. Banerjee and C. Peikert, "New and improved key-homomorphic pseudorandom functions," in *Proc. CRYPTO. 2014*, pp. 353–370.
[78] Z. Li, H. Zhu, G. Liao, M. Wang, P. Li, and P. Gope, "Qt-pake: Secure messaging via quantum-safe threshold pake," *IEEE Trans. Consum. Electron.*, 2025.
[79] R. Yang, M. H. Au, J. Lai, Q. Xu, and Z. Yu, "Lattice-based techniques for accountable anonymity: composition of abstract stern's protocols and weak prf with efficient protocols from lwr," *Cryptology ePrint Archive*, 2017.
[80] M. Albrecht and K. D. Gur, "Verifiable oblivious pseudorandom functions from lattices: Practical-ish and thresholdisable," in *Proc. ASIACRYPT 2024*, pp. 205–237.
[81] D. Wang and P. Wang, "On the implications of zipf's law in passwords," in *Proc. ESORICS 2016*, pp. 111–131.
[82] J. Jiang, D. Wang, and G. Zhang, "Qpause: Quantum-resistant password-protected data outsourcing for cloud storage," *IEEE Trans. Serv. Comput.*, vol. 17, no. 3, pp. 1140–1153, 2024.
[83] F. Benhamouda and D. Pointcheval, "Verifier-based password-authenticated key exchange: New models and constructions," Cryptology ePrint Archive, Paper 2013/833, 2013. [Online]. Available: https://eprint.iacr.org/2013/833
[84] Q. Wang, D. Wang, C. Cheng, and D. He, "Quantum2fa: efficient quantum-resistant two-factor authentication scheme for mobile devices," *IEEE Trans. Depend. Secur. Comput.*, vol. 20, no. 1, pp. 193–208, 2021.
[85] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *Proc. EUROCRYPT 2000*, pp. 139–155.
[86] J. Jiang, D. Wang, G. Zhang, and Z. Chen, "Quantum-resistant password-based threshold single-sign-on authentication with updatable server private key," in *Proc. ESORICS 2022*, pp. 295–316.
[87] E. Di Giandomenico, Y. Li, and S. Schäge, "Protoss: Protocol for tight optimal symmetric security," in *Proc. ACM CCS 2024*, pp. 4718–4731.
[88] D. Wang and P. Wang, "Two birds with one stone: Two-factor authentication with security beyond conventional bound," *IEEE Trans. Depend. Secur. Comput.*, vol. 15, no. 4, pp. 708–722, 2016.
[89] L. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. ACM STOC 1996*, pp. 212–219.
[90] C. Gentry, P. MacKenzie, and Z. Ramzan, "Password authenticated key exchange using hidden smooth subgroups," in *Proc. ACM CCS 2005*, pp. 299–309.
[91] M. Abdalla and D. Pointcheval, "Simple password-based encrypted key exchange protocols," in *Proc. CT-RSA 2005*, pp. 191–208.
[92] J. Katz, R. Ostrovsky, and M. Yung, "Efficient password-authenticated key exchange using human-memorable passwords," in *Proc. EUROCRYPT 2001*.
[93] P. Das, H. Julia, and L. Anja., "Dpase: distributed password-authenticated symmetric-key encryption, or how to get many keys from one password," in *Proc. AsiaCCS 2022*, pp. 682–696.
[94] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. ACM CCS 1993*.
[95] D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry, "Random oracles in a quantum world," in *Proc. ASIACRYPT 2011*, pp. 41–69.
[96] S. Hohenberger, S. Myers, R. Pass, and abhi shelat, "ANONIZE: A large-scale anonymous survey system," in *Proc. IEEE S&P 2014*, pp. 375–389.



Jingwei Jiang received his Ph.D. degree in the College of Computer Science and Technology, Harbin Engineering University, China, in 2024. Currently, he is an assistant researcher at Nankai University. As the first author, he has published papers at TIFS 2024, TSC 2023, ACM CSUR 2025, ESORICS 2022, the Chinese Journal of Computers, ICICS 2025, etc. His research interests include lattice-based cryptography and authentication.



Ding Wang received his Ph.D. degree in Information Security at Peking University in 2017, and was supported by the "Boya Postdoctoral Fellowship" in Peking University from 2017 to 2019. Currently, he is a Full Professor at Nankai University. As the first author (or corresponding author), he has published more than 140 papers at venues like IEEE S&P, ACM CCS, NDSS, USENIX Security, IEEE TDSC and IEEE TIFS. His research has been reported by over 200 media like The Wall Street Journal, Daily Mail, Forbes, IEEE Spectrum, and Communications of the ACM, appeared in the Elsevier 2017 "Article Selection Celebrating Computer Science Research in China", and resulted in the revision of the authentication guideline NIST SP800-63. He has been involved in the community as a PC Chair/TPC member for over 60 international conferences such as USENIX Security 2027, NDSS 2023-2025, ACM CCS 2022, PETS 2022-2024, ACSAC 2020-2024, ACM AsiaCCS 2027/2025/2022/2021, FC 2025-2027, IFIP SEC 2018-2026, ICICS 2018-2026, and SPNCE 2020-2022. He has received the "ACM China Outstanding Doctoral Dissertation Award", the Best Paper Award at INSCRYPT 2018, the First Prize of Cryptologic Innovation Award of China Association for Cryptologic Research, and the First Prize of Natural Science Award of Ministry of Education. His research interests focus on passwords, authentication, and provable security.