Universally Composable Hybrid PAKE Secure Against Harvest-Now-Decrypt-Later Attacks

Yasmine Vazirinejad[1], Feng Hao[1], You Lyu[2], and Shengli Liu[2]

[1] University of Warwick, Coventry, UK
{yasmine.vazirinejad,feng.hao}@warwick.ac.uk
[2] Shanghai Jiao Tong University, Shanghai, China
{vergil,slliu}@sjtu.edu.cn

Abstract. We present a hybrid password-authenticated key exchange (PAKE) protocol that is secure against harvest-now-decrypt-later (HNDL) attacks by quantum adversaries, and is universally composable under the parallel composition framework of Lyu and Liu (EUROCRYPT 2025). Existing hybrid PAKE constructions combine a classical PAKE with a post-quantum (PQ) PAKE, with the overall security intended to rely on the stronger of the two. However, identifying which PAKE is stronger is non-trivial, given the limited maturity of post-quantum PAKE designs. Recognizing that the immediate quantum threat is passive, we propose a different hybrid compiler: rather than combining two PAKEs, we encapsulate a classical PAKE within a standard post-quantum Key Encapsulation Mechanism (KEM). This modular separation avoids the fragility of post-quantum password handling while neutralizing HNDL attacks. Our compiler works with any two-pass or three-pass PAKEs. As a concrete instantiation, we construct a three-pass protocol that combines J-PAKE and a post-quantum KEM. We also implement the resulting protocol and provide performance results demonstrating that the hybrid construction remains practical, with the complete handshake executing in 2.81 ms. This construction has the distinctive advantage that it does not require any ideal cipher, (constant-time) hash-to-curve, or trusted setup assumptions. Within the Lyu-Liu framework, we show that J-PAKE satisfies the notion of a *Full DH-type PAKE*. We model the KEM as a password-independent *Simulatable DH-type* component satisfying the minimal simulation properties required for parallel composition. To capture the prospective quantum threat, we formalize a stronger variant of the standard HNDL threat model—where the quantum adversary is explicitly granted the plaintext password—and prove that our protocol achieves *Session Key Security* and *Post-Quantum Forward Secrecy*. Our construction relies solely on standardized and widely deployed primitives, yielding a hybrid PAKE that is UC-secure, efficient, and well-suited for real-world deployment during the post-quantum transition.

Keywords: Hybrid PAKE · UC-Security · J-PAKE · KEM · PQ Transtion

1 Introduction

Password-Authenticated Key Exchange (PAKE) represents a category of protocols that allow two remote parties to establish a shared key based on a shared password without relying on any public-key infrastructure. Since the seminal work of Bellovin and Merritt [12], PAKE has become a foundational cryptographic technique and is widely deployed in real-world systems, including electronic passports, the IoT commissioning process, wireless authentication, secure messaging, and client–server authentication protocols [23].

Classical PAKEs and DH-Type Constructions. A large class of practical PAKEs is based on Diffie–Hellman–style constructions, where the password is used to authenticate or mask group exponentiation without enabling offline dictionary attacks. Early protocols such as EKE [12,11] and SPEKE [28] introduced the paradigm of password-aided Diffie–Hellman. Subsequent designs refined this approach by eliminating idealized assumptions and strengthening security guarantees, including PAK and PPK [15], SPAKE2 [6,36], Dragonfly/SAE [25], and CPace [22]. Most PAKE protocols require an ideal cipher, a (constant-time) hash-to-curve function or a trusted setup [23], which are delicate to implement correctly. J-PAKE is one of the few PAKE designs that does not require these, and has been standardized (in ISO/IEC and Thread) and deployed widely [23].

From a structural point of view, many of these protocols can be modeled as DH-type PAKEs, where parties exchange group elements and derive a shared Diffie–Hellman secret that is subsequently combined with the password through a key-derivation function. A recent work of Lyu and Liu [34] has shown that a broad subclass of such protocols, called full DH-type PAKEs, admits simulation-based security proofs in the Universal Composability (UC) framework. This abstraction captures classical two-round schemes such as SPAKE2 and TBPEKE [38], provided that suitable simulation and extraction properties hold.

Post-Quantum PAKEs and the Maturity Gap. The advent of quantum computing threatens classical PAKEs based on discrete logarithms, as Shor's algorithm will eventually break the Diffie–Hellman assumption. In response, several post-quantum PAKEs have been proposed, primarily relying on lattice-based or isogeny-based assumptions. These constructions generally fall into three categories: *(i) EKE-style PAKEs* (e.g., RLWE-PPK [18], CAKE [10]), which replace classical Diffie–Hellman with post-quantum KEMs and encrypt flows under the password; *(ii) OEKE-style and KEM-based PAKEs* (e.g., EKE-KEM [39], CHIC [8], OQUAKE [42]), which rely on strong uniformity or anonymity properties of the KEM; and *(iii) Modular constructions* employing smooth projective hash functions or oblivious transfer [35,3,19,20,29,30,16].

While these works represent major progress toward the post-quantum era, purely post-quantum PAKEs are not yet mature enough for wide-scale deployment and standardization [7]. Entangling low-entropy passwords with noisy post-quantum ciphertexts (such as LWE) is complex, and they often necessitate idealizations like the Quantum Random Oracle Model (QROM) or the Ideal Cipher Model (ICM). Furthermore, many fully post-quantum designs incur substantial

bandwidth overhead, require trusted setups, or require non-standard properties from the underlying primitives that standardized algorithms do not provide.

The Immediate Threat: HNDL and the Case for a Hybrid Transition. Because large-scale quantum computers do not yet exist, an active quantum adversary capable of real-time offline dictionary attacks is not a present reality. The imminent, practical threat is *harvest-now, decrypt-later* (HNDL). In this scenario, an adversary records classical PAKE transcripts today with the intent of breaking the session key once quantum technology matures. Fully post-quantum designs seek to protect both the session key and the password against future quantum adversaries, but these constructions have not yet reached the maturity and deployment confidence of classical PAKEs. No post-quantum PAKE protocols have been standardized or deployed in practice. This motivates the design of hybrid PAKE by combining a classic PAKE and a post-quantum PAKE, in case the post-quantum scheme fails. The security of the hybrid design is intended to rely on the stronger PAKE, but which is the stronger PAKE remains unclear with the current state of knowledge.

Hybrid PAKEs and the Challenge of Composition. Recent progress on hybrid PAKE has been driven by two independent frameworks: the PAKE combiner approach of Hesse and Rosenberg [26], and the DH-type PAKE framework of Lyu and Liu [34]. Although both works aim to achieve hybrid security in the UC framework, they differ in their modeling assumptions and in the conditions imposed on parallel composition.

Hesse and Rosenberg study generic PAKE combiners and analyze both sequential and parallel composition. For parallel composition, their framework imposes a strong requirement on *both* PAKE components: namely, that they satisfy *statistical password hiding* (SPH). Informally, SPH requires that the transcript distribution of the protocol be information-theoretically independent of the password, even if all computational assumptions underlying the protocol are broken. Under this strict definition, Hesse and Rosenberg show that only a narrow class of classical PAKEs (such as SPAKE2 [6] and CPace [22]) qualify for secure parallel composition. By contrast, J-PAKE does not satisfy SPH, as a quantum attacker can extract the password from the transcript (e.g., by computing discrete logarithms x_4 and $x_4 \cdot pw$ in the third pass to obtain pw; see Table 3). However, we observe that the SPH definition does not fully capture the reality of offline dictionary attacks. When the hardness assumptions fail (as assumed in SPH), an adversary can bypass transcript-level protections in a SPH-compliant protocol (e.g., SPAKE2 and CPace): they need only derive the session key for each dictionary candidate and verify it against captured encrypted traffic or key confirmation messages. Through this *offline search*, the adversary can break the password and the session key. Therefore, the advantage of the SPH property for a PAKE protocol is limited against a quantum adversary.

Lyu and Liu take a structurally different approach by abstracting PAKEs as DH-type protocols equipped with explicit simulation algorithms. Their framework introduces the notion of a *Full DH-type PAKE*, characterized by perfect simulation of protocol messages, one-wayness of the derived DH key, and unique

password extraction under active attacks. In their analysis of parallel composition, they show that the parallel execution of two DH-type components is secure as long as *one* component satisfies the Full DH-type properties, while the other satisfies a weaker set of simulation properties (namely, perfect simulability of messages and derived keys). Crucially, their security proof is phrased entirely in terms of simulation. As a consequence, their parallel composition theorem does not require both of the components to satisfy statistical password hiding; the security guarantees are inherited from the Full DH-type component, ensuring the Universal Composability of the entire construction.

This distinction explains the necessity of the Lyu-Liu framework for our construction. Under the Hesse-Rosenberg definition, a parallel hybrid PAKE involving J-PAKE would be impossible due to the lack of SPH. Under the Lyu-Liu framework, however, we show that parallel composition is viable provided that the post-quantum component can be modeled as a DH-type protocol with simulatable transcripts and the classical component as a full DH-type PAKE.

Concrete Hybrid Instantiations. While the frameworks above provide the theoretical foundation for composition, only a few concrete hybrid PAKE instantiations have been proposed in the literature. Vos et al. [42] proposed CPaceOQUAKE, a serial composition of CPace and a specific post-quantum scheme OQUAKE. While efficient, this construction imposes non-standard requirements on the underlying KEM, such as anonymity and uniformity, and results in a relaxed security guarantee that leaks to passive observers whether passwords match. Separately, Lyu and Liu [34] suggested several instantiations within their DH-type framework. These include a serial composition of SPAKE2 [6] with the lattice-based scheme CHIC [8], which relies on the strong ideal cipher model (ICM), and a parallel composition of SPAKE2 with the isogeny-based (Crs)X-GA-PAKE [4,27], which requires a trusted setup (CRS). They also noted a serial composition of SPAKE2 with PAKE^{QRO} [35] for QROM security, though this variant requires more rounds, trusted setups, and similarly incurs password equality leakage.

Fundamentally, the theoretical goal of a hybrid PAKE is indeed to remain secure against active quantum adversaries as long as the post-quantum component remains secure [34]. However, to formally prove this security, the post-quantum component must be proven in a model that grants the adversary quantum capabilities, such as the Quantum Random Oracle Model (QROM). If the post-quantum component only has a proof in a classical model, the hybrid protocol is not yet formally proven secure against an active quantum attacker [7].

When applying this criterion to evaluate existing hybrid constructions, the SPAKE2 + PAKE^{QRO} [35] composition is the only hybrid among the listed designs that achieves provable full security against active quantum adversaries, as its post-quantum component is proven in the QROM. The other hybrid constructions rely on classical idealized models such as the standard ROM or the Ideal Cipher Model (ICM). Because these classical models do not account for quantum superposition queries, they do not provide proven cryptographic guar-

antees against an active quantum attacker attempting to exploit those specific idealizations.

1.1 Technical Overview and Contributions

A DH-Type PAKE with a Parallel KEM Layer. In this work, we adopt a perspective that aligns with the realities of the post-quantum transition. Our starting point is the three-pass J-PAKE protocol [24], which is run in parallel with a post-quantum KEM to protect the second and third passes. The first pass is *password-independent* and therefore naturally free from any possible password leakage against a quantum attacker. HNDL attacks are prevented because the second and third protocol messages are encrypted under the post-quantum KEM, preventing a transcript-harvesting adversary from recovering the password-dependent data needed to perform an offline dictionary attack.

To support UC analysis, we first extend the Full DH-type PAKE definitions to explicitly accommodate three-pass protocols. We then show that J-PAKE satisfies this strengthened definition, achieving UC security under standard classical assumptions. On top of this, we add a post-quantum KEM layer, instantiated with ML-KEM (Kyber) [14]. Crucially, this KEM layer has no password input. The initiator generates a pair of ephemeral keys and passes her public key to the responder, where he generates a high-entropy shared secret. Both parties then use that secret to encrypt the second and third passes. From the perspective of hybrid composition, this KEM layer behaves like a simulatable DH-type component. As a result, it fits naturally into the parallel composition framework of Lyu-Liu, allowing us to prove hybrid security without requiring SPH for the classical layer. We compare our schemes with other hybrid PAKEs in Table 1.

We focus on J-PAKE as our concrete instantiation because its first pass is password-independent, making it naturally resistant to attacks on the password outside the KEM layer. However our hybrid PAKE framework can serve as a generic compiler that can be applied to any DH-type PAKE protocol. For any generic 3-pass DH-type PAKE, the application is direct: the first pass transmits the KEM public key alongside password-independent data, while the subsequent password-dependent flows are KEM-encrypted. Furthermore, our compiler accommodates 2-pass PAKEs (such as EKE, CPace, and SPAKE2) to yield an optimal, strict 2-pass hybrid protocol. Because the first flow in a 2-pass PAKE is necessarily password-dependent, applying the KEM simultaneously requires that the classical component protects the password in the clear. This is achieved by inheriting the specific setup assumptions of the underlying classical protocol—namely, an ideal cipher (EKE), constant-time hash-to-curve (CPace), or a trusted setup/CRS (SPAKE2) [23] (see Table 1). Moreover, by prepending a single pass to establish the KEM public key, any 2-pass PAKE can be generically transformed into a 3-pass hybrid PAKE with HNDL security like our concrete construction. In practice, trading a 2-pass structure for a 3-pass protocol to avoid these assumptions is a minor cost, since PAKE is typically used only once for initial key establishment, with subsequent session keys derived from pre-shared key (PSK) methods as in TLS 1.3.

Table 1: Comparison of Hybrid PAKEs. **CRS**: Common Reference String. **ICM**: Ideal Cipher Model. **ROM**: Random Oracle Model. **QROM**: Quantum Random Oracle Model. **HNDL**: Harvest-Now-Decrypt-Later. **AGM**: Algebraic Group Model.

Protocol	Hybrid Type	C1 Assumption (Classical)	C2 Assumption (Post-Quantum)	Security	Model	Passes	No hash-to-curve	No trusted setup	No ideal cipher	HNDL security
Existing Hybrid Designs (Classical PAKE + PQ PAKE)										
CPaceOQUAKE [42]	Serial	Gap-CDH	MLWE OW-PCA ANO-PCA Uniform public keys Special encoding mapping	UC (Relaxed) $\mathcal{F}_{\mathsf{PAKE}^{\mathsf{Re}}}$	ROM	3		✓	✓	✓
SPAKE2 + CHIC [34]	Serial	Gap-CDH	LWE Lattice	UC $\mathcal{F}_{\mathsf{PAKE}}$	ICM CRS	3	✓			✓
SPAKE2 + (Crs)X-GA-PAKE [34]	Parallel	Gap-CDH	Isogeny GA-GapCDH DSim-GA-GapCDH	UC $\mathcal{F}_{\mathsf{PAKE}}$	ROM CRS	2	✓		✓	✓
SPAKE2 + $\mathsf{PAKE}^{\mathsf{QRO}}$ [34,35]	Serial	Gap-CDH	LWE / Isogeny GA-GapCDH	UC $\mathcal{F}_{\mathsf{PAKE}}$	QROM CRS	4	✓		✓	✓
Our Compiler & Instantiations (Classical PAKE + PQ KEM)										
EKE + ML-KEM	**Parallel**	**CDH** [33]	**MLWE** **IND-CCA2**	**UC** $\mathcal{F}_{\mathsf{PAKE}}$	**ICM**	**2**	✓	✓		✓
SPAKE2 + ML-KEM	**Parallel**	**Gap-CDH** [34]	**MLWE** **IND-CCA2**	**UC** $\mathcal{F}_{\mathsf{PAKE}^{\mathsf{LE}}}$	**CRS**	**2**	✓		✓	✓
CPACE + ML-KEM	**Parallel**	**sCDH** **sSDH** **sTCDH** [5]	**MLWE** **IND-CCA2**	**UC** $\mathcal{F}_{\mathsf{PAKE}^{\mathsf{LE}}}$ [3]	**ROM**	**2**		✓	✓	✓
J-PAKE + ML-KEM	**Parallel**	**Gap-CDH** **DDH**	**MLWE** **IND-CCA2**	**UC** $\mathcal{F}_{\mathsf{PAKE}^{\mathsf{LE}}}$	**ROM** **AGM**	**3**	✓	✓	✓	✓

To illustrate this structural generality, Figure 1 provides a high-level overview of both the 3-pass and 2-pass generic compilers. In these diagrams, the classical PAKE layer is highlighted in purple, while the post-quantum KEM layer is highlighted in green. The abstract functions PAKE.Init and PAKE.Resp represent the generic message generation algorithms of the underlying classical protocol, producing the classical payload flows (denoted as m_1, m_2, m_3), which are finally evaluated by PAKE.Key to extract the classical key K_1. While our formal UC proofs in subsequent sections model these components specifically as DH-type PAKEs to leverage the Lyu-Liu parallel composition framework [34], the encapsulation architecture itself is agnostic. As shown via the boxed payloads ($\boxed{m_i}$), the compiler treats the classical messages as black-box strings, wrapping them within the KEM's symmetric encryption layer (Enc_{ss}) whenever possible, before binding both independent secrets (K_1 and ss) together with the transcript (τ) into the final session key via a key derivation function.

Contributions. In this work, we present a practical and theoretically sound transition path tailored to the immediate threat facing PAKEs. The contributions are summarized as follows.

[3] The exact description of the "lazy-extraction" functionality of CPACE is explained in [5].

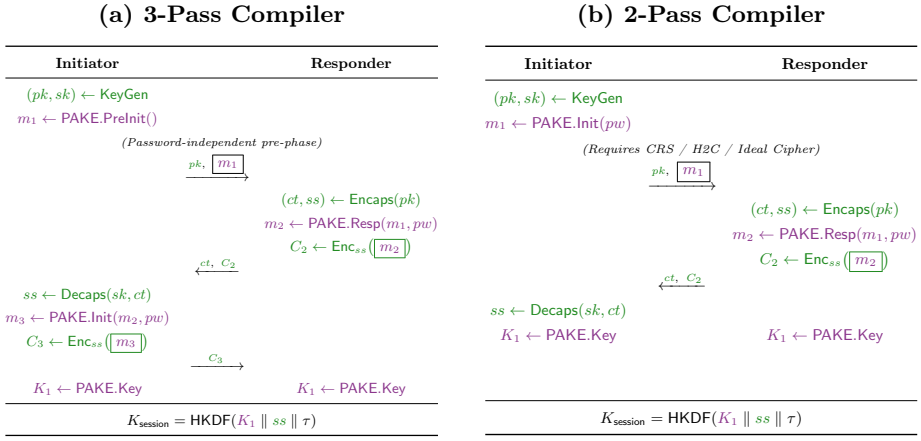

Fig. 1: Structural overview of the generic compilers. **(a) 3-Pass:** By utilizing a password-independent first pass, all password-dependent classical flows ($\boxed{m_2}$, $\boxed{m_3}$) are encapsulated within the KEM layer's symmetric encryption, eliminating the need for extra assumptions. **(b) 2-Pass:** The first password-dependent flow ($\boxed{m_1}$) is exposed in the clear. To maintain security against passive harvesting, the underlying classical protocol must protect the password without KEM assistance, necessitating setup assumptions like Hash-to-Curve, a CRS, or an Ideal Cipher.

- We propose a novel hybrid PAKE design that combines a classical PAKE and a post-quantum KEM via parallel hybrid composition without extra rounds, using only standardized primitives. Compared with other hybrid designs, our approach preserves the well-established security of a classical PAKE against active classical adversaries, with additional security against HNDL attacks as well as post-quantum forward secrecy. We present a concrete instantiation by combining J-PAKE and Kyber. Provided benchmark results demonstrate the practicality and efficiency of the resulting hybrid protocol, with a complete handshake requiring only 2.81 ms.
- We show that J-PAKE, when modeled as a three-pass DH-type PAKE, constitutes a *Full DH-type PAKE* with computational simulation properties and achieves UC security under standard classical assumptions. Furthermore, the post-quantum KEM (Kyber) layer can be modeled as a password-independent DH-type component that admits perfect simulation of messages and pseudorandomness of keys.
- We formally analyze our hybrid construction in the UC framework using a PAKE functionality closely assigned with that of Lyu-Liu [34]. The parallel composition of the classical PAKE and post-quantum KEM components yields a hybrid PAKE that inherits UC security from the classical component. Complementing this, we formalize a stronger version of the HNDL

attack model and prove that our protocol achieves *Session Key Security* and *Post Quantum Forward Secrecy* against future quantum adversaries.

2 Preliminaries

2.1 PAKE

A Password-Authenticated Key Exchange (PAKE) protocol enables two parties sharing a low-entropy password to establish a high-entropy session key over an untrusted network. Recent work by Lyu and Liu [34] formalizes a broad subclass of practical protocols as *DH-type PAKEs*, abstracting the structure of password-aided Diffie–Hellman computations.

Definition 1 (2-round DH-type PAKE [34]). *A 2-round DH-type PAKE consists of five algorithms:* $\mathsf{DHSetup}$ *outputs public parameters;* $\mathsf{DHInit}(pw)$ *and* $\mathsf{DHResp}(pw)$ *generate password-dependent messages and secret states for the initiator and responder, respectively; and deterministic algorithms* Compi *and* Compr *compute a shared DH-key based on the password, secret states, and exchanged messages.*

To model our construction, it is necessary to distinguish between a "round" and a "pass". The classical notion of *round complexity* counts the number of alternating dependency stages, often assuming that both parties can start simultaneously and may need to wait for messages before continuing computation. In our work, we use the term *pass* or *flow* to refer to a single, unidirectional message transmission. The 2-round DH-type formulation above assumes a structure where parties can generate their initial messages simultaneously. While a "round" can technically encompass one or two flows, this abstraction implicitly assumes that the first-round messages depend only on the password and local randomness. However, this abstraction is structurally insufficient for J-PAKE [24], which does not use any password in the first round. For ease of description in the hybrid setting, we consider J-PAKE to follow a sequential three-pass structure (Initiator $\rightarrow$ Responder $\rightarrow$ Initiator). In this structure, we explicitly model the initial flow as password-free to capture the setup of ephemeral parameters (like X_1, X_2 in J-PAKE) before the password is utilized. The responder's subsequent password-dependent message relies on the initiator's initial commitment, and the initiator's final confirmation relies on the responder's reply. Forcing this flow into a simultaneous two-round abstraction would obscure the specific sequential dependencies required for our simulation. Therefore, we extend the Lyu-Liu framework by formalizing a **3-pass DH-type PAKE** definition below.

Definition 2 (3-pass DH-type PAKE). *A 3-pass DH-type PAKE scheme is defined by the tuple of algorithms* $\Pi = (\mathsf{DHSetup}, \mathsf{PreInit}, \mathsf{PreResp}, \mathsf{DHResp}, \mathsf{DHInit}, \mathsf{Compi}, \mathsf{Compr})$ *with the following syntax:*

- $\mathsf{DHSetup}(1^\kappa)$*: On input the security parameter* 1^κ*, outputs public parameters* pp *and a trapdoor* td*. All other algorithms implicitly take* pp *as input.*

– $\mathsf{PreInit}()$*: Run by the initiator* ***without*** *the password, it outputs an initial message* $m_0^{(I)}$ *and a state* $st_0^{(I)}$*.*
– $\mathsf{PreResp}()$*: Run by the responder* ***without*** *the password, it outputs an initial message* $m_0^{(R)}$ *and a state* $st_0^{(R)}$*.*
– $\mathsf{DHResp}(pw, st_0^{(R)}, m_0^{(I)})$*: On input a password* pw*, the responder's pre-state* $st_0^{(R)}$*, and the initiator's initial message* $m_0^{(I)}$*, outputs a second-pass message* m_2 *and a state* $st_1^{(R)}$*.*
– $\mathsf{DHInit}(pw, st_0^{(I)}, m_0^{(R)}, m_2)$*: On input a password* pw*, the initiator's pre-state* $st_0^{(I)}$*, and the responder's second-pass message* $m_0^{(R)}$ *and* m_2*, outputs a third-pass message* m_1 *and a state* $st_1^{(I)}$*.*
– $\mathsf{Compi}(pw, st_0^{(I)}, st_1^{(I)}, m_0^{(I)}, m_0^{(R)}, m_1, m_2)$*: Deterministic algorithm run by the initiator to output a DH-key* K_I*.*
– $\mathsf{Compr}(pw, st_0^{(R)}, st_1^{(R)}, m_0^{(I)}, m_0^{(R)}, m_1, m_2)$*: Deterministic algorithm run by the responder to output a DH-key* K_R*.*

Finally, both parties derive the session key as $\mathsf{Key}_{\mathsf{session}} = \mathsf{HKDF}(pw, K_i, \tau, id, id')$*, where* $i \in \{I, R\}$*,* $K_I = K_R$*, and the transcript* τ *includes all public messages of the protocol.*

Correctness. A 3-pass DH-type PAKE satisfies perfect correctness if an honest execution of the protocol always yields matching keys. Formally, for any password pw and public parameters $(pp, td) \leftarrow \mathsf{DHSetup}(1^\kappa)$, if the initiator and responder honestly execute the protocol algorithms $(\mathsf{PreInit}, \mathsf{PreResp}, \mathsf{DHResp}, \mathsf{DHInit})$ to produce their respective secret states and a public transcript $(m_0^{(I)}, m_0^{(R)}, m_1, m_2)$, then applying the deterministic derivation functions yields $\Pr[\mathsf{Compi}(\dots) = \mathsf{Compr}(\dots)] = 1$.

Table 2 illustrates the 3-pass PAKE definition, where the first pass is password-independent, and the password-dependent computations are split across two later passes.

2.2 Key Encapsulation Mechanism

A KEM enables two parties to establish a shared secret key using asymmetric cryptography. It is defined in the following.

Definition 3 (Key Encapsulation Mechanism). *A KEM scheme* $\Pi_{\mathsf{KEM}} = (\mathsf{KeyGen}, \mathsf{Encaps}, \mathsf{Decaps})$ *over a key space* $\mathcal{K}$ *is defined as follows:*

– $\mathsf{KeyGen}(1^\kappa)$*: A probabilistic algorithm that outputs a public/secret key pair* (pk, sk)*.*
– $\mathsf{Encaps}(pk)$*: A probabilistic algorithm that takes a public key* pk *and outputs a ciphertext* c *and a shared secret* $K \in \mathcal{K}$*.*
– $\mathsf{Decaps}(sk, c)$*: A deterministic algorithm that takes a secret key* sk *and a ciphertext* c*, and outputs a shared secret* $K \in \mathcal{K}$ *or a rejection symbol* $\perp$*.*

Table 2: Definition of the 3-pass DH-type PAKE protocol Π. The transcript τ includes $\{m_0^{(I)}, m_0^{(R)}, m_1, m_2\}$. Initiator is known as Party P and responder as CP. The sub-session id ($ssid$) is defined as $\{\tau \| id \| id'\}$ for both parties.

Initiator (i, id, pw)		**Responder** (j, id', pw)
Input: pp, pw, sid, id, id'		
Setup: $(pp, td) \leftarrow \mathsf{DHSetup}(1^\kappa)$		
Pass 1: Pre-Phase		
$(m_0^{(I)}, st_0^{(I)}) \leftarrow \mathsf{PreInit}()$	$\xrightarrow{m_0^{(I)}, id}$	
		$(m_0^{(R)}, st_0^{(R)}) \leftarrow \mathsf{PreResp}()$
Pass 2: Responder Binding		
		$(m_2, st_1^{(R)}) \leftarrow \mathsf{DHResp}(pw, st_0^{(R)}, m_0^{(I)})$
	$\xleftarrow{m_0^{(R)}, m_2, id'}$	
Pass 3: Initiator Binding		
$(m_1, st_1^{(I)}) \leftarrow \mathsf{DHInit}(pw, st_0^{(I)}, m_0^{(R)}, m_2)$		
	$\xrightarrow{m_1}$	
Key Derivation		
$K_I \leftarrow \mathsf{Compi}(pw, st_{0,1}^{(I)}, \tau)$		$K_R \leftarrow \mathsf{Compr}(pw, st_{0,1}^{(R)}, \tau)$
$\mathsf{Key}_{\mathsf{session}} := \mathsf{HKDF}(pw, K, \tau, id, id')$		
Output		
$\mathsf{K}_{\mathsf{session}}, id', ssid$		$\mathsf{K}_{\mathsf{session}}, id, ssid$

Correctness. A KEM is $(1-\delta)$-correct if for all $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa)$: $\Pr[\mathsf{Decaps}(sk, c) = K \mid (c, K) \leftarrow \mathsf{Encaps}(pk)] \geq 1 - \delta$.

For our instantiation with ML-KEM/Kyber [14], correctness holds with negligible error. We rely on its standard IND-CCA2 security (See Definition 14, Appendix B), which is proven in the Random and Quantum Random Oracle Model (ROM and QROM) under the MLWE assumption against adversaries making at most q_H queries.

2.3 Universal Composable Framework

Password-authenticated key exchange (PAKE) protocols are commonly analyzed using simulation-based security in the universally composable (UC) framework. In the UC framework, security is defined by comparing two execution models: the *real world* and the *ideal world*. In the real world, parties execute the protocol Π over a network controlled by an adversary $\mathcal{A}$, observed by an environment $\mathcal{Z}$. In the ideal world, dummy parties interact with a trusted ideal functionality $\mathcal{F}$, and the adversary is replaced by a simulator Sim, whose goal is to emulate the real-

world adversary's behavior using only the interface provided by $\mathcal{F}$. A protocol Π *securely realizes* an ideal functionality $\mathcal{F}$ if, for every real-world adversary $\mathcal{A}$, there exists a simulator Sim such that for every environment $\mathcal{Z}$, the executions are computationally indistinguishable: $\forall \mathcal{A}\ \exists \mathsf{Sim}\ \forall \mathcal{Z}\ :\ \mathsf{Real}^{\Pi}_{\mathcal{A},\mathcal{Z}} \approx \mathsf{Ideal}^{\mathcal{F}}_{\mathsf{Sim},\mathcal{Z}}$. Figure 2 illustrates these execution models.

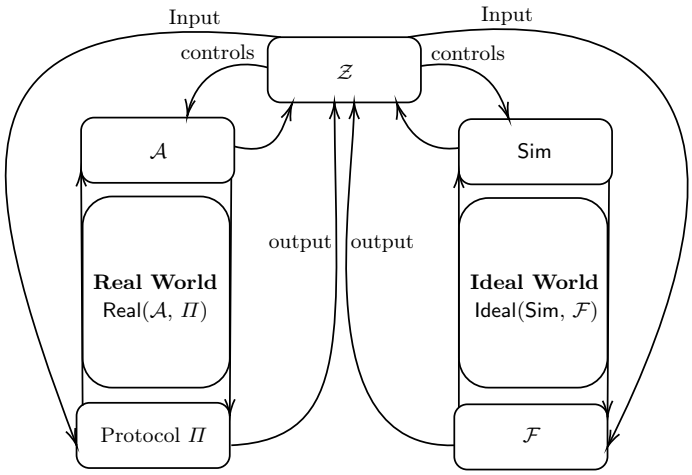

Fig. 2: UC Security Intuition: the environment $\mathcal{Z}$ interacts with the real world $(\mathcal{A}, \Pi)$ or with the ideal world $(\mathsf{Sim}, \mathcal{F})$, and must not be able to distinguish the two.

2.4 Ideal Functionality of PAKE

The security guarantees of a PAKE protocol depend on the choice of the ideal functionality $\mathcal{F}$. In this work, we adopt the *Bare PAKE with Lazy-Extraction* functionality, denoted as $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$ [9]. The formal definition is provided in Appendix A. This functionality closely aligns with the choice of Liu and Lyu [34]. We choose this functionality for two primary theoretical advantages:

1. **Accurate Threat Modeling (Lazy-Extraction):** The original $\mathcal{F}_{\mathsf{PAKE}}$ [17] precludes offline dictionary attacks entirely, while "leaky" variants [40] unrealistically notify the adversary if their guess succeeds. Lazy-Extraction [1] models the silent nature of real-world offline attacks by permitting the adversary a single offline guess after key generation without explicit success notification.
2. **Modular Design (Bare PAKE):** The "Bare" variant [9] simplifies identifiers by eliminating the requirement for pre-shared sub-session identifiers

(*ssid*). Parties input only a global *sid*, their *id*, and password, drastically reducing implementation hurdles and explicitly allowing state reuse.

From Bare to Full Functionality. While $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$ simplifies the analysis, real-world applications require explicit session binding. As proven in prior work [9,34], any protocol securely realizing $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$ can be transformed to realize the standard multi-session functionality $\mathcal{F}_{\mathsf{PAKE}^{\mathsf{LE}}}$ by binding the password to the session context: $pw' := \mathcal{H}(sid \parallel ssid \parallel C \parallel CP \parallel pw)$.

In our security analysis, we prove that the J-PAKE component is a *Full DH-type PAKE*, which implies secure emulation of $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$ [34]. By applying this transformation, our hybrid protocol achieves standard multi-session security. Finally, as noted in [1], protocols realizing $\mathcal{F}_{\mathsf{PAKE}^{\mathsf{LE}}}$ with mutual authentication imply perfect forward secrecy in the indistinguishability-based model.

2.5 Harvest-Now-Decrypt-Later (HNDL) Security

The standard notion of forward secrecy considers a passive attacker who compromises long-term keys in the future. However, classical forward secrecy relies on classical hardness assumptions and does not account for the retrospective breakage of these assumptions by future quantum adversaries. Recent standardization efforts emphasize the practical relevance of the Harvest-Now-Decrypt-Later (HNDL) threat model, where an adversary passively records protocol transcripts today and exploits future quantum computational advances to compromise past sessions.

Our hybrid design does not aim to add to the password privacy already provided by the underlying classical PAKE. Instead, it adds value by strengthening the *session key* against a quantum attacker. By ensuring the session key remains secure even if the classical assumptions fall, this design strengthens forward secrecy, which prevents HNDL attacks.

We formalize this by defining HNDL-Session Key Security. While a standard HNDL model typically only grants the adversary the transcript, we define a *stronger* variant that simultaneously captures *Post-Quantum Perfect Forward Secrecy*:

> Even if a future quantum adversary learns the correct password *pw and* completely breaks the classical PAKE transcript, the final session key remains computationally indistinguishable from random due to the password-independent post-quantum KEM layer.

Definition 4 (Strong HNDL-Session Key Security). *Let Π be a PAKE protocol. We define the HNDL-Session Key Security of Π against a quantum adversary $\mathcal{A}_Q$ via the experiment $\mathsf{Exp}^{\mathsf{HNDL-SKS}}_{\Pi,\mathcal{A}_Q}(\kappa)$:*

1. ***Execution (Harvesting):** The Challenger runs an honest execution of Π between two parties using the correct password pw. This yields a real transcript τ and a real session key K_1.*

2. ***Challenge Phase:*** *The Challenger selects a bit* $b \leftarrow \{0,1\}$.
   - *If* $b = 1$ (***Real Key***)*: The Challenger sets the challenge key* $K_b = K_1$.
   - *If* $b = 0$ (***Random Key***)*: The Challenger samples a key* $K_0 \leftarrow \$\mathcal{K}$ *uniformly at random and sets* $K_b = K_0$.
3. ***Observation (Post-Quantum Analysis):*** *The Challenger gives* $\mathcal{A}_Q$ *the tuple* (τ, K_b, pw).
   Note that $\mathcal{A}_Q$ *is an adversary capable of breaking classical assumptions (e.g., discrete logarithms) post-execution.*
4. ***Guess:*** $\mathcal{A}_Q$ *outputs a bit* b'.

The advantage is defined as $\mathsf{Adv}^{\mathsf{HNDL-SKS}}_{\Pi,\mathcal{A}_Q}(\kappa) = |\Pr[b' = 1] - \Pr[b' = 0]|$. *We say* Π *achieves* ***Strong HNDL-Session Key Security*** *if this advantage is negligible for any* $\mathcal{A}_Q$*, provided the post-quantum KEM assumption holds.*

3 The Hybrid PAKE Components

In this section, we formalize the two building blocks required for our hybrid construction: the J-PAKE protocol (Section 3.1) and a post-quantum Key Encapsulation Mechanism (KEM) layer (Section 3.2).

The core intuition behind our choice of components is to leverage the parallel composition framework of Lyu and Liu [34]. By modeling J-PAKE as a *Full DH-type PAKE* with *Computational Simulation properties* and the KEM layer as a *Simulatable DH-type component*, we construct a protocol where the session key is bound to both primitives. This design ensures that the hybrid protocol inherits UC security from the classical component, while the KEM layer provides an independent layer of post-quantum confidentiality. Unlike tightly coupled designs where the password is used to mask post-quantum ciphertexts, our construction maintains a clear separation of concerns: The password ensures authentication against active classical adversaries, while the KEM ensures that the session key remains indistinguishable from random to passive quantum adversaries. Thus, even if the classical assumptions are eventually compromised, the confidentiality of the established session key is preserved.

As the UC security is inherited from the classical component, our compiler can be applied to any UC-secure PAKE, such as CPace [22] or SPAKE2 [6]. Note that SPAKE2 was already proven to be a *Full DH-type* PAKE by [34] and inherently achieves UC security. In this work, we use J-PAKE [24] as our concrete instantiation. We chose J-PAKE because it relies purely on standard assumptions, naturally accommodates the 3-pass transformation, and crucially, allows us to formally prove its UC security.

3.1 J-PAKE Component

J-PAKE is a symmetric protocol and is defined as a two-round PAKE [24]. Without loss of generality and based on Definition 2, we consider it a three-pass scheme with Alice initiating the communication, as shown in Table 3.

Table 3: The Original J-PAKE protocol from [2] and [24] and the revised 3-pass J-PAKE based on Table 2 and Definition 2.

Original J-PAKE		**Revised 3-pass J-PAKE**	
Alice	**Bob**	**Initiator**	**Responder**
$(\mathbb{G}, g, p, q, H, \Sigma_{\text{NIZK}}) \leftarrow \mathsf{DHSetup}(1^\kappa)$		$(\mathbb{G}, g, p, q, H, \Sigma_{\text{NIZK}}) \leftarrow \mathsf{DHSetup}(1^\kappa)$	
Round 1: Exchange		*Pass 1: Pre-Phase*	
$x_1 \leftarrow \mathbb{Z}_q$,	$x_3 \leftarrow \mathbb{Z}_q$,	$x_1 \leftarrow \mathbb{Z}_q$,	$x_3 \leftarrow \mathbb{Z}_q$,
$x_2 \leftarrow \mathbb{Z}_q^*$,	$x_4 \leftarrow \mathbb{Z}_q^*$,	$x_2 \leftarrow \mathbb{Z}_q^*$,	$x_4 \leftarrow \mathbb{Z}_q^*$,
$X_1 = g^{x_1}$,	$X_3 = g^{x_3}$,	$X_1 = g^{x_1}$,	$X_3 = g^{x_3}$,
$\pi_1 \leftarrow \mathsf{ZKP}(x_1)$,	$\pi_3 \leftarrow \mathsf{ZKP}(x_3)$,	$\pi_1 \leftarrow \mathsf{ZKP}(x_1)$,	$\pi_3 \leftarrow \mathsf{ZKP}(x_3)$,
$X_2 = g^{x_2}$,	$X_4 = g^{x_4}$,	$X_2 = g^{x_2}$,	$X_4 = g^{x_4}$,
$\pi_2 \leftarrow \mathsf{ZKP}(x_2)$	$\pi_4 \leftarrow \mathsf{ZKP}(x_4)$	$\pi_2 \leftarrow \mathsf{ZKP}(x_2)$	$\pi_4 \leftarrow \mathsf{ZKP}(x_4)$
$\xleftrightarrow{X_3, X_4, \pi_3, \pi_4 \mid X_1, X_2, \pi_1, \pi_2}$		$\xrightarrow{X_1, X_2, \pi_1, \pi_2}$	
Round 2: Binding		*Pass 2: Binding*	
Verify (π_3, π_4),	Verify (π_1, π_2),		Verify (π_1, π_2)
Abort if $X_4 = 1$	Abort if $X_2 = 1$		Abort if $X_2 = 1$
$A = (X_1 X_3 X_4)^{x_2 \cdot pw}$	$B = (X_1 X_2 X_3)^{x_4 \cdot pw}$		$B = (X_1 X_2 X_3)^{x_4 \cdot pw}$
$\pi_A \leftarrow \mathsf{ZKP}(x_2 \cdot pw)$	$\pi_B \leftarrow \mathsf{ZKP}(x_4 \cdot pw)$		$\pi_B \leftarrow \mathsf{ZKP}(x_4 \cdot pw)$
$\xleftrightarrow{B, \pi_B \mid A, \pi_A}$		$\xleftarrow{X_3, X_4, \pi_3, \pi_4, B, \pi_B}$	
Computation		*Pass 3: Computation*	
Verify π_B	Verify π_A	Verify (π_3, π_4, π_B)	
$K = (B/g^{x_2 x_4 pw})^{x_2}$	$K = (A/g^{x_2 x_4 pw})^{x_4}$	Abort if $X_4 = 1$	
		$A = (X_1 X_3 X_4)^{x_2 \cdot pw}$	
		$\pi_A \leftarrow \mathsf{ZKP}(x_2 \cdot pw)$	
		$\xrightarrow{A, \pi_A}$	
			Verify (π_A)
		$K_1 = (B/g^{x_2 x_4 pw})^{x_2}$	$K_1 = (A/g^{x_2 x_4 pw})^{x_4}$
		$= g^{(x_1 + x_3) x_2 x_4 pw}$	$= g^{(x_1 + x_3) x_2 x_4 pw}$

Based on the definition of the 3-pass DH-type PAKE (Table 2), the final key derivation explicitly includes the password pw, the session context $\{id, id'\}$, and the protocol transcript. This inclusion is necessary to satisfy the state-extraction requirements of the UC functionality $\mathcal{F}_{\mathsf{bPAKE}}^{\mathsf{LE}}$. In practical deployments, the implicit password authentication of the DH-key K_1 is often sufficient, though retaining context identifiers is recommended for secure session binding.

With the protocol structure of Table 3, we now formalize the security requirements. To apply the UC proof of the full DH-type PAKEs theorem of Lyu and Liu [34] to our setting, we extend their notion of a *Full DH-type PAKE* to three-pass protocols. The following definition mandates that the protocol admits efficient simulation of all messages and keys, a necessary condition for proving UC security.

Definition 5 (Full 3-pass DH-type PAKE: with simulators). *A 3-pass PAKE is a full DH-type PAKE if it is a DH-type one consisting of the algorithms:* DHSetup, PreInit, PreResp, DHInit, DHResp, Compi *and* Compr *also, there exist the following simulation algorithms:* SimPreInit, SimPreResp, SimDHInit, SimDHResp, SimCompi *and* SimCompr. *The simulation algorithms are defined as follows:*

- SimPreInit(pp)*: Takes as input the public parameter pp, and outputs a pre-phase message and state* $(m_0^{(I)}, st_0^{(I)})$.
- SimPreResp(pp)*: Takes as input the public parameter pp, and outputs a pre-phase message and state* $(m_0^{(R)}, st_0^{(R)})$.
- SimDHResp$(pp, st_0^{(R)}, m_0^{(I)})$*: Takes as input pp, the simulated (or real) responder pre-state* $st_0^{(R)}$*, and the initiator's pre-phase message* $m_0^{(I)}$*; outputs a second-pass message and state* $(m_2, st_1^{(R)})$.
- SimDHInit$(pp, st_0^{(I)}, m_0^{(R)}, m_2)$*: Takes as input pp, the simulated (or real) initiator pre-state* $st_0^{(I)}$*, and the responder's pre-phase message* $m_0^{(R)}$ *and* m_2*; outputs a third-pass message and state* $(m_1, st_1^{(I)})$.
- SimCompi$(td, pw, st_0^{(I)}, st_1^{(I)}, m_0^{(I)}, m_0^{(R)}, m_1, m_2)$*: Takes as input the trapdoor td, password pw, the initiator's states* $(st_0^{(I)}, st_1^{(I)})$*, and all public messages* $(m_0^{(I)}, m_0^{(R)}, m_1, m_2)$*, and outputs a DH-key* K.
- SimCompr$(td, pw, st_0^{(R)}, st_1^{(R)}, m_0^{(I)}, m_0^{(R)}, m_1, m_2)$*: Takes as input* (td, pw)*, the responder's states* $(st_0^{(R)}, st_1^{(R)})$*, and all public messages* $(m_0^{(I)}, m_0^{(R)}, m_1, m_2)$*, and outputs a DH-key* K.

In the ideal world of a UC framework, a simulator Sim reproduces the view of $\mathcal{Z}$ without access to the passwords by interacting with an ideal PAKE functionality $\mathcal{F}$. To achieve UC security, the simulator must be able to: (i) simulate all protocol transcripts under both passive and active attacks without knowing the password; (ii) generate pseudorandom session keys for passive attacks and unsuccessful active attacks using $\mathcal{F}$; and (iii) extract the password and maintain key consistency for successful active attacks using the single password-testing capability provided by $\mathcal{F}$. The above requirements are captured by the defining properties of a full DH-type PAKE protocol, which we define below.

Definition 6 (One-Wayness of the DH-Key). *For any PPT adversary* $\mathcal{A}$ *with oracle access to* $\mathcal{O}_1, \mathcal{O}_2$*, its advantage* $\mathsf{Adv}^{\mathsf{ow}}_{\mathsf{key}}(\mathcal{A})$ *is negligible, where*

$\mathsf{Adv}^{\mathsf{ow}}_{\mathsf{key}}(\mathcal{A})$ *is defined as:*

$$\Pr\left[\begin{array}{l}(pp, td) \leftarrow \mathsf{DHSetup};\ pw \leftarrow \mathcal{A}(pp);\ (m_0^{(I)}, st_0^{(I)}) \leftarrow \mathsf{PreInit}();\\ (m_0^{(R)}, st_0^{(R)}) \leftarrow \mathsf{PreResp}(); (m_2, st_1^{(R)}) \leftarrow \mathsf{DHResp}(pw, st_0^{(R)}, m_0^{(I)});\\ (m_1, st_1^{(I)}) \leftarrow \mathsf{DHInit}(pw, st_0^{(I)}, m_0^{(R)}, m_2);\\ K \leftarrow \mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2}(pp, pw, m_0^{(I)}, m_0^{(R)}, m_1, m_2) :\\ \qquad K = \mathsf{Compi}(pw, st_0^{(I)}, st_1^{(I)}, m_0^{(I)}, m_0^{(R)}, m_1, m_2)\end{array}\right].$$

The checking oracles for query inputs $(pw', K', st_0, st_1, m_0^{(I)}, m_0^{(R)}, m_1, m_2)$ *are defined inline as* $\mathcal{O}_1(\dots) := [K' \stackrel{?}{=} \mathsf{Compi}(pw', st_0^{(I)}, st_1^{(I)}, m_0^{(I)}, m_0^{(R)}, m_1, m_2)]$ *and* $\mathcal{O}_2(\dots) := [K' \stackrel{?}{=} \mathsf{Compr}(pw', st_0^{(R)}, st_1^{(R)}, m_0^{(I)}, m_0^{(R)}, m_1, m_2)]$*, utilizing the initiator and responder states, respectively. In other words, given a full transcript, it is infeasible for an adversary to compute the DH key* K*. Consequently, in the random-oracle model, the derived session key* $\mathsf{Key} = H(pw, m_0^{(I)}, m_0^{(R)}, m_1, m_2, K)$ *is indistinguishable from uniform in passive executions and failed active attacks, allowing the simulator to sample a random session key to satisfy requirement (ii).*

Definition 7 (Computational Simulation for the Initiator). *For all* $(pp, td) \leftarrow \mathsf{DHSetup}$*, all passwords* pw*, and all fixed responder messages* $(m_0^{(R)}, m_2)$*, the real execution view is computationally indistinguishable from the simulated view:*

$$\underbrace{(m_0^{(I)}, m_0^{(R)}, m_1, m_2, K)}_{\mathsf{Real}} \approx_c \underbrace{(m_0'^{(I)}, m_0^{(R)}, m_1', m_2, K')}_{\mathsf{Simulated}}$$

where the Real *tuple is generated by the honest algorithms* $(\mathsf{PreInit}, \mathsf{DHInit}, \mathsf{Compi})$*, and the* Simulated *tuple is generated by the simulator algorithms* $(\mathsf{SimPreInit}, \mathsf{SimDHInit}, \mathsf{SimCompi})$ *using the trapdoor* td*. In other words, for any PPT distinguisher* $\mathcal{D}$*, the advantage in distinguishing these distributions is negligible:* $\mathsf{Adv}^{\mathsf{Init}}_{\mathsf{Sim}}(\mathcal{D}) := |\Pr[\mathcal{D}(\mathsf{Real}) = 1] - \Pr[\mathcal{D}(\mathsf{Simulated}) = 1]| \leq \mathsf{negl}(\kappa)$.

Definition 8 (Computational Simulation for the Responder). *For all* $(pp, td) \leftarrow \mathsf{DHSetup}$*, all passwords* pw*, and all fixed initiator messages* $(m_0^{(I)}, m_1)$*, the real execution view is computationally indistinguishable from the simulated view:*

$$\underbrace{(m_0^{(I)}, m_0^{(R)}, m_1, m_2, K)}_{\mathsf{Real}} \approx_c \underbrace{(m_0^{(I)}, m_0'^{(R)}, m_1, m_2', K')}_{\mathsf{Simulated}}$$

where the Real *tuple is generated by the honest algorithms* $(\mathsf{PreResp}, \mathsf{DHResp}, \mathsf{Compr})$*, and the* Simulated *tuple is generated by the simulator algorithms* $(\mathsf{SimPreResp}, \mathsf{SimDHResp}, \mathsf{SimCompr})$ *using the trapdoor* td*. In other words, for any PPT distinguisher* $\mathcal{D}$*, the advantage in distinguishing these distributions is negligible:* $\mathsf{Adv}^{\mathsf{Resp}}_{\mathsf{Sim}}(\mathcal{D}) := |\Pr[\mathcal{D}(\mathsf{Real}) = 1] - \Pr[\mathcal{D}(\mathsf{Simulated}) = 1]| \leq \mathsf{negl}(\kappa)$.

For the computational simulation property, all protocol messages, including the pre-phase, DHInit and DHResp messages, must be simulatable with at least a **computationally indistinguishable distribution** without knowledge of the password. In particular, simulators SimPreInit, SimPreResp, SimDHInit, and SimDHResp allow the simulator to generate a complete 3-pass transcript $(m_0^{(I)}, m_0^{(R)}, m_1, m_2)$ independently of pw. This enables simulation of both passive and active executions against any efficient (PPT) adversary and fulfills requirement (i).

Remark 1 (Relaxation to Computational Simulatability). We note that the original definition of a Full DH-type PAKE by Lyu and Liu [34] stipulates *perfect simulation* for protocol messages. While satisfied by some protocols, J-PAKE relies on the Decisional Diffie-Hellman (DDH) assumption for transcript privacy, satisfying *computational simulatability*. This relaxation is fully compatible with the modular UC framework; it implies that the indistinguishability of the real and ideal world is bounded by the advantage of a DDH adversary rather than being information-theoretically zero, which remains negligible.

Definition 9 (Unique Password for the Initiator). *Let* $\tau^* = (m_0^{(I)*}, m_0^{(R)}, m_1^*, m_2)$ *be the simulated transcript and* $\sigma^* = (st_0^{(I)*}, st_1^{(I)*})$ *be the simulated initiator states. The advantage* $\mathsf{Adv}^{\mathsf{Init}}_{\mathsf{Unipw}}(\mathcal{A})$ *is the probability that* $\mathcal{A}$ *outputs two valid keys for distinct passwords, defined as:*

$$\Pr\left[\begin{array}{l}(pp^*, td^*) \leftarrow \mathsf{DHSetup};\ (m_0^{(I)*}, st_0^{(I)*}) \leftarrow \mathsf{SimPreInit}(pp^*);\\ (m_1^*, st_1^{(I)*}) \leftarrow \mathsf{SimDHInit}(pp^*, st_0^{(I)*}, m_0^{(R)}, m_2);\\ (pw, pw', m_0^{(R)}, m_2, K, K') \leftarrow \mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_1^*}(pp^*, m_0^{(I)*}, m_1^*) :\\ \qquad pw \neq pw' \ \wedge\ K = \mathsf{SimCompi}(td^*, pw, \sigma^*, \tau^*) \wedge\\ \qquad\qquad K' = \mathsf{SimCompi}(td^*, pw', \sigma^*, \tau^*)\end{array}\right].$$

The checking oracles for query inputs $(td^*, pw', st_0, st_1, m_0^{(I)}, m_0^{(R)}, m_1, m_2, K')$ *are defined as* $\mathcal{O}_1(\dots) := [K' \stackrel{?}{=} \mathsf{SimCompi}(td^*, pw', st_0^{(I)}, st_1^{(I)}, m_0^{(I)}, m_0^{(R)}, m_1, m_2)]$, $\mathcal{O}_2(\dots) := [K' \stackrel{?}{=} \mathsf{SimCompr}(td^*, pw', st_0^{(R)}, st_1^{(R)}, m_0^{(I)}, m_0^{(R)}, m_1, m_2)]$, *and* $\mathcal{O}_1^*(\dots) := [K' \stackrel{?}{=} \mathsf{SimCompi}(td^*, pw', \sigma^*, \tau^*)]$. *For input oracle* $\mathcal{O}_1$, *the states are for the initiator; for oracle* $\mathcal{O}_2$, *the states are for the responder; and for oracle* $\mathcal{O}_1^*$, *the inputs will be* $(td^*, pw', \sigma^*, \tau^*, K')$.

Definition 10 (Unique Password for the Responder). *Let* $\tau^* = (m_0^{(I)}, m_0^{(R)*}, m_1, m_2^*)$ *be the simulated transcript and* $\sigma^* = (st_0^{(R)*}, st_1^{(R)*})$ *be the simulated responder states. The advantage* $\mathsf{Adv}^{\mathsf{Resp}}_{\mathsf{Unipw}}(\mathcal{A})$ *is the probability that* $\mathcal{A}$

outputs two valid keys for distinct passwords, defined as:

$$\Pr\left[\begin{array}{l}(pp^*, td^*) \leftarrow \mathsf{DHSetup};\ (m_0^{(I)}, st_0^{(I)}) \leftarrow \mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_2^*}(pp^*);\\ (m_0^{(R)*}, st_0^{(R)*}) \leftarrow \mathsf{SimPreResp}(pp^*);\\ (m_2^*, st_1^{(R)*}) \leftarrow \mathsf{SimDHResp}(pp^*, st_0^{(R)*}, m_0^{(I)});\\ (pw, pw', st_1^{(I)}, m_1, K, K') \leftarrow \mathcal{A}^{\mathcal{O}_1,\mathcal{O}_2,\mathcal{O}_2^*}(pp^*, m_0^{(I)}, m_2^*, st_0^{(I)}, st_0^{(R)*}) :\\ \qquad pw \neq pw' \ \wedge\ K = \mathsf{SimCompr}(td^*, pw, \sigma^*, \tau^*) \ \wedge\\ \qquad K' = \mathsf{SimCompr}(td^*, pw', \sigma^*, \tau^*)\end{array}\right].$$

The checking oracles for query inputs $(td^*, pw', st_0, st_1, m_0^{(I)}, m_0^{(R)}, m_1, m_2, K')$ *are defined as* $\mathcal{O}_1(\dots) := [K' \stackrel{?}{=} \mathsf{SimCompi}(td^*, pw', st_0, st_1, m_0^{(I)}, m_0^{(R)}, m_1, m_2)]$, $\mathcal{O}_2(\dots) := [K' \stackrel{?}{=} \mathsf{SimCompr}(td^*, pw', st_0, st_1, m_0^{(I)}, m_0^{(R)}, m_1, m_2)]$, *and* $\mathcal{O}_2^*(\dots) := [K' \stackrel{?}{=} \mathsf{SimCompr}(td^*, pw', \sigma^*, \tau^*)]$. *For* $\mathcal{O}_1$, *the states are for the initiator; for* $\mathcal{O}_2$, *the states are for the responder; and for* $\mathcal{O}_2^*$, *the inputs are* $(td^*, pw', \sigma^*, \tau^*, K')$.

For the unique password property, for successful active attacks, the simulator must extract the unique correct password consistent with the transcript. This is achieved via: (i) perfect DH-key simulation, ensuring that SimCompi and SimCompr faithfully reproduce the real DH key for any adversarially chosen messages; and (ii) the unique-password property for both initiator and responder, which guarantees that no two distinct passwords yield the same simulated DH key. As a result, incorrect password guesses are rejected, while the correct password can be uniquely extracted from hash queries. In failed active attacks, the adversary never queries the hash on the correct password and the session key remains uniform, fulfilling requirement (ii). In successful attacks, the simulator extracts the correct password and programs the session key accordingly, fulfilling requirement (iii).

From definition 2, the 3-pass version of J-PAKE [24] is a DH-type PAKE with the mappings based on Table 7, provided in Appendix C. For proving that it also satisfy Definitions 6 to 10, we extend the mappings to Table 8, Appendix C and prove the following lemmas. The proof of the following lemmas appear in the Appendix D, Subsection D.1, D.2, D.3, D.4 and D.5, respectively. Proving these lemmas then will result in proving the UC security of J-PAKE in Theorem 1, which we provide the proof in Appendix D.6.

Unlike the original analysis of J-PAKE, which relies on the CDH and DDH assumptions, our UC proof requires the Gap-CDH and DDH assumptions to satisfy the Full DH-type PAKE definition. Specifically, Gap-CDH enables the simulator to answer adaptive key-validity queries during the One-Wayness reduction (Lemma 1) via a DDH oracle, which standard CDH cannot support. We separately retain the DDH assumption to ensure the computational indistinguishability of simulated transcripts.

Lemma 1 (One-Wayness of the DH-key for 3-pass J-PAKE). *Let* $\Pi_{\mathsf{J-PAKE}}$ *be the 3-pass J-PAKE protocol of Table 3 instantiated in a cyclic*

group $\mathbb{G} = \langle g \rangle$ of prime order q, with simulation-sound extractable NIZKs [21]. Assume the hardness of Gap-CDH *[41] and* DDH *in $\mathbb{G}$. Then for every PPT adversary $\mathcal{A}$ with oracle access to $\mathcal{O}_1, \mathcal{O}_2$ as in Definition 6, we have* $\mathsf{Adv}^{\mathsf{ow}}_{\mathsf{key}}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{Gap\text{-}CDH}}(\mathcal{B}) + \mathsf{Adv}_{\mathsf{DDH}}(\mathcal{A}') + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIZK}}(\mathcal{A}'') = \mathsf{negl}(\kappa)$.

Lemma 2 (Computational Simulation for the Initiator). *Assume the hardness of the* DDH *problem in $\mathbb{G}$ and that the underlying NIZK system is Zero-Knowledge. Then, for all $(pp, td) \leftarrow$ DHSetup, all passwords $pw \in \mathbb{Z}_q^*$, and all fixed responder messages $(m_0^{(R)}, m_2)$, the real transcript distribution is* ***computationally indistinguishable*** *from the simulated transcript: $(m_0^{(I)}, m_0^{(R)}, m_1, m_2, K) \approx_c (m_0'^{(I)}, m_0^{(R)}, m_1', m_2, K')$, where the real and simulated variables are defined as in Definition 7. Concretely, for any PPT distinguisher $\mathcal{D}$, there exist PPT algorithms $\mathcal{D}'$ and $\mathcal{D}''$ such that the advantage is bounded by* $\mathsf{Adv}^{\mathsf{Init}}_{\mathsf{Sim}}(\mathcal{D}) \leq \mathsf{Adv}_{\mathsf{DDH}}(\mathcal{D}') + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIZK}}(\mathcal{D}'')$.

Lemma 3 (Computational Simulation for the Responder). *J–PAKE satisfies Definition 8: the distributions of $(m_0^{(I)}, m_0^{(R)}, m_1, m_2, K)$ and $(m_0^{(I)}, m_0'^{(R)}, m_1, m_2', K')$ are computationally indistinguishable. Concretely, for any PPT distinguisher $\mathcal{D}$, there exist PPT algorithms $\mathcal{D}'$ and $\mathcal{D}''$ such that the advantage is bounded by* $\mathsf{Adv}^{\mathsf{Resp}}_{\mathsf{Sim}}(\mathcal{D}) \leq \mathsf{Adv}_{\mathsf{DDH}}(\mathcal{D}') + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIZK}}(\mathcal{D}'')$.

Lemma 4 (Unique Password for the Initiator). *Let $\Pi_{\mathsf{J\text{-}PAKE}}$ be the 3-pass J-PAKE protocol of Table 3 in a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q. Assume the ZK proofs are simulation-sound extractable NIZKs (SE-NIZK) [21], and let $(pp^*, td^*) \leftarrow$ DHSetup where td^* is the SE-NIZK extraction key. Assume the hardness of* Gap-CDH *and* DDH *in $\mathbb{G}$. Then for every PPT adversary $\mathcal{A}$ with oracle access to $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_1^*$ as in Definition 9,* $\mathsf{Adv}^{\mathsf{Init}}_{\mathsf{Unipw}}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{Gap\text{-}CDH}}(\mathcal{B}) + \mathsf{Adv}_{\mathsf{DDH}}(\mathcal{A}') + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIZK}}(\mathcal{A}'') = \mathsf{negl}(\kappa)$.

Lemma 5 (Unique Password for the Responder). *J–PAKE satisfies Definition 10: For the simulated responder state, there exists at most one password consistent with the simulated key under all responder oracles.*

Theorem 1 (UC Security of 3-pass J-PAKE). *Let Π be the 3-pass J-PAKE protocol described in Table 3. Assume the hardness of* Gap-CDH *and* DDH *in $\mathbb{G}$ and the existence of Simulation-Sound Extractable NIZKs (SE-NIZK) in the Random Oracle Model. We have proven that Π satisfies all five properties of a* Full DH-type PAKE *as defined in Definitions 6–10. Therefore, by Theorem 1 of Lyu and Liu [34], for every PPT adversary $\mathcal{A}$, there exists a simulator* Sim *such that for every environment $\mathcal{Z}$ $\mathsf{Real}^{\Pi}_{\mathcal{A},\mathcal{Z}} \approx_c \mathsf{Ideal}^{\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}}_{\mathsf{Sim},\mathcal{Z}}$ Hence, Π UC-emulates the bare lazy-extraction PAKE functionality $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$ with a tight bound of:*

$$\mathsf{Adv}^{\mathsf{UC}}_{\mathcal{Z}} \leq \underbrace{\ell^2 \cdot \mathsf{Adv}^{\mathsf{ow}}_{\mathsf{key}}}_{\textit{Passive Security}} + \underbrace{\ell \cdot \left(\mathsf{Adv}^{\mathsf{Init}}_{\mathsf{Unipw}} + \mathsf{Adv}^{\mathsf{Resp}}_{\mathsf{Unipw}}\right)}_{\textit{Active Extraction}} + \underbrace{\ell(\mathsf{Adv}^{\mathsf{Init}}_{\mathsf{Sim}} + \mathsf{Adv}^{\mathsf{Resp}}_{\mathsf{Sim}})}_{\textit{Transcript Simulatability}}.$$

Corollary 1 (Standard Multi-Session UC Security [34]). *Applying the standard password transformation $pw' := \mathcal{H}(P \parallel CP \parallel sid \parallel ssid \parallel pw)$ to the*

protocol Π (where P is the current party, CP the counterparty, sid the global session identifier, and ssid the sub-session identifier) securely lifts this emulation to the full multi-session functionalities $\mathcal{F}_{\mathsf{PAKE}^{\mathsf{LE}}}$ in the Random Oracle Model.

3.2 The KEM Component

Having established the security of the J-PAKE component, we now formalize the second building block of our hybrid construction: the post-quantum Key Encapsulation Mechanism (KEM) layer. This layer is responsible for establishing a post-quantum shared secret and symmetrically encrypting the second and third passes of the protocol transcript.

We model this component as a *Simulatable DH-type Protocol*. Unlike the classical J-PAKE component, the KEM layer is *password-independent*; it does not take pw as an input. However, it can still be cast into the 3-pass DH-type syntax (Definition 2) via two minor structural simplifications: First, the password input is explicitly ignored. Second, because standard KEMs only require the initiator to generate a public key, the responder's pre-phase algorithm ($\mathsf{PreResp}$) is omitted, returning empty values for $m_0^{(R)}$ and $st_0^{(R)}$. This simplification adheres to the protocol definition, as the responder's encapsulation ciphertext naturally forms the second-pass message m_2, alongside the updated state $st_1^{(R)}$.

In this construction, the DH-Key is exactly the KEM shared secret ($K_2 = ss$). The ephemeral public key of the initiator serves as the pre-phase message, while the KEM ciphertexts and symmetrically encrypted payloads constitute the subsequent flows. Table 9 in Appendix C shows the algorithmic mapping table of the KEM layer into the DH-type framework.

Table 4 illustrates the execution flow. To maintain modularity for the parallel composition proof, this component outputs raw entropy (ss). Similar to the J-PAKE component, the cryptographic binding of this secret to the session identifiers and the password is performed in the final Hybrid Protocol wrapper.

Although the KEM component is not a *Full* DH-type PAKE (as it lacks password-dependent properties like uniqueness), it satisfies the simulation requirements for parallel composition. Specifically, we prove that it admits **Perfect Simulation** (denoted by $\equiv$), meaning the simulated transcript distribution is mathematically identical to the real execution. This meets and exceeds the relaxed Computational Simulation requirement (denoted by $\approx_c$) utilized for the J-PAKE component.

According to Theorem 2 of Lyu and Liu [34], a parallel hybrid PAKE remains secure as long as one component is Full DH-type (satisfied by J-PAKE), while the auxiliary components need only be *statistically simulatable*. The KEM component satisfies this requirement trivially: because it does not utilize the password as an input, the simulation algorithms simply execute the honest protocol algorithms using freshly sampled randomness. We formalize these required properties in Lemmas 6 and 7, and their proofs in Appendix D, subsection D.7 and D.8, respectively.

Table 4: Execution flow of the KEM Component. The setup phase generates global parameters pp. Parameters are based on ML-KEM [14] over the ring R_q.

Input: $pp^{\mathsf{KEM}} = (R_q, k, \eta) \leftarrow \mathsf{DHSetup}^{\mathsf{KEM}}(1^\kappa)$

Initiator (Alice)	**Responder (Bob)**
Pass 1: Pre-Phase (Key Exchange)	
$(PK_E^A, SK_E^A) \leftarrow \mathsf{KeyGen}$	
Set $m_0^{(I)} := PK_E^A$	
$\xrightarrow{m_0^{(I)}}$	
Pass 2: Responder Encapsulation	
	$(ct_2, ss) \leftarrow \mathsf{Encaps}(PK_E^A)$
	$C_2 \leftarrow \mathsf{Enc}_{ss}(X_3, X_4, B, \pi_B, \pi_3, \pi_4)$
	Set $m_2^{KEM} := ct_2, C_2$
$\xleftarrow{m_2^{KEM}}$	
Pass 3: Initiator Decapsulation	
$ss \leftarrow \mathsf{Decaps}(SK_E^A, ct_2)$	
$(X_3, X_4, B, \pi_B, \pi_{3,4}) \leftarrow \mathsf{Dec}_{ss}(C_2)$	
$C_3 \leftarrow \mathsf{Enc}_{ss}(A, \pi_A)$	
Set $m_1^{KEM} := C_3$	
$\xrightarrow{m_1^{KEM}}$	
	$(A, \pi_A) \leftarrow \mathsf{Dec}_{ss}(C_3)$
Key Computation	
$K_2 = (ss)$	$K_2 = (ss)$

Lemma 6 (Perfect Simulation for the Initiator in the KEM Component). *Let* $(pp^{\mathsf{KEM}}, \perp) \leftarrow \mathsf{DHSetup}^{\mathsf{KEM}}(1^\kappa)$. *For all passwords* pw, *all fixed responder messages* $(m_0^{(R)} := \perp, m_2)$, *and any fixed payload string* pl, *the transcript distribution generated by the simulator is identical to the real execution:*

$$(m_0^{(I)}, m_1, m_2, K_2) \mid_{\mathsf{payload}=\mathsf{pl}} \equiv (m_0^{\prime(I)}, m_1', m_2, K_2') \mid_{\mathsf{payload}=\mathsf{pl}}$$

Lemma 7 (Perfect Simulation for the Responder in the KEM Component). *For all passwords* pw *and all fixed initiator messages* $(m_0^{(I)}, m_1)$, *and any fixed payload string* pl *the following distributions are identical:* $(m_0^{(I)}, m_1, m_2, K_2) \mid_{\mathsf{payload}=\mathsf{pl}} \equiv (m_0^{(I)}, m_1, m_2', K_2') \mid_{\mathsf{payload}=\mathsf{pl}}$.

3.3 Computational Security of the KEM Component

While we have established that the KEM component admits perfect simulation of its structural messages (by running honest algorithms), we must now show that this simulated view provides the necessary semantic security. Specifically, we demonstrate that the encrypted J-PAKE payloads remain confidential and that the derived KEM keys are computationally indistinguishable from random strings. Establishing these properties is essential for guaranteeing post-quantum forward secrecy. This ensures that even if the classical component is eventually compromised, the KEM-derived secrets remain secure against passive quantum adversaries, thereby defeating Harvest-Now-Decrypt-Later (HNDL) attacks. The proof of lemma 8 appears in Appendix D, subsection D.9.

Lemma 8 (Pseudorandomness and Ciphertext Indistinguishability). *Let* $\mathsf{View}^{\mathsf{Real}}$ *denote the view of an adversary in a real execution, where ciphertexts encrypt real J-PAKE payloads and* K_2 *is the derived shared secret. Let* $\mathsf{View}^{\mathsf{Sim}}$ *denote the view generated by a simulator that encrypts dummy payloads (strings of zeros) using a uniformly random key* $K_2 \leftarrow \$\mathcal{K}$*. Assume the KEM is IND-CCA2 secure (implied by MLWE) and the symmetric encryption scheme (SE) is IND-CPA secure. Then for any PPT distinguisher* $\mathcal{D}$*, operating as an active classical network adversary during the real-time execution phase, the advantage is bounded by:* $\left|\Pr[\mathcal{D}(\mathsf{View}^{\mathsf{Real}}) = 1] - \Pr[\mathcal{D}(\mathsf{View}^{\mathsf{Sim}}) = 1]\right| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}}(\mathcal{B}_1) + 2 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SE}}(\mathcal{B}_2) = \mathsf{negl}(\kappa)$.

4 The Hybrid Protocol

In this section, we present the full design of our concrete Hybrid Password-Authenticated Key Exchange protocol. Our construction integrates the two components analyzed previously:

1. **The Classical Layer:** A Full DH-type PAKE derived from J-PAKE (Section 3.1), providing active security and password authentication.
2. **The Post-Quantum Layer:** A Simulatable DH-type protocol instantiated with ML-KEM (Section 3.2), providing long-term confidentiality.

The design objective is to achieve *hybrid security*: the protocol must remain secure as long as at least one of the two underlying hardness assumptions holds. Specifically, we target two distinct guarantees:

– **Password Privacy (UC Security):** Guaranteed by the classical layer (Gap-CDH and DDH). If this layer breaks, password privacy is lost.
– **Session Confidentiality:** Guaranteed by the post-quantum layer (MLWE). Even if the classical layer breaks by an active classical adversary, the session key remains indistinguishable from random.

4.1 High-Level Structure

Table 5 details the complete protocol execution. The interaction consists of three passes:

- **First Flow (Alice → Bob):** Alice acts as the Initiator. She generates her ephemeral KEM keypair (PK_E^A, SK_E^A) and sends the public key PK_E^A in the clear, alongside her initial J-PAKE message $m_0^{(I)} = (X_1, X_2, \pi_1, \pi_2)$.
- **Second Flow (Bob → Alice):** Bob encapsulates a fresh secret ss against Alice's key PK_E^A. He generates his J-PAKE message $m_0^{(R)} = (X_3, X_4, \pi_3, \pi_4)$. Crucially, Bob encrypts his J-PAKE payload (including the second-round value B) using the KEM-derived secret ss. He sends the KEM ciphertext ct_2 and the encrypted payload C_2 to Alice.
- **Third Flow (Alice → Bob):** Alice decapsulates ss to decrypt Bob's payload. She then encrypts her final J-PAKE message $m_1 = (A, \pi_A)$ using the same ss. She sends the encrypted payload C_3 to Bob.

The KEM secret (ss) contributes to the post-quantum entropy of the final key. The J-PAKE component computes the classical shared value K_1, and the final hybrid key is constructed via $K_{\mathsf{session}} = \mathsf{HKDF}(K_1 \| ss \| \tau)$ where τ is the $\mathsf{transcript}$ and includes the real PK_E^A and all other public messages. For the UC security proof, the table must follow the consistency of Table 2; it must explicitly mix the password pw into the HKDF along with the parties' id as well. For implementing the protocol, the final session key can be written as the concatenation of both components' keys and the initiator's ephemeral public key.

4.2 Rationale for Encryption in Flows 2 and 3

Unlike the original J-PAKE which sends DH items with ZKPs in the clear, our hybrid construction encrypts the protocol transcript using the ephemeral KEM key. This satisfies two critical security goals:

(i) Real-time password authentication. The password authentication happens in real time during the key exchange process. Therefore, an active attacker has only a short window to solve discrete logarithms, which are required to break the security of J-PAKE. Successful password authentication during the key exchange effectively rules out an active quantum adversary (since a quantum computer is not available as of today).

(ii) Long-term data secrecy. By wrapping the classical transcript in a post-quantum KEM, we ensure that a passive quantum adversary recording the traffic today cannot decrypt the data in the future, even with a quantum computer. This preserves long-term data secrecy, even in the post-quantum era.

Table 5: The Hybrid PAKE Protocol (Π_{HPAKE}). Purple denotes J-PAKE elements, Green denotes KEM elements. Adding the context to the final session key is for the theoretical UC proof. π_i denotes the zero-knowledge proof for variable i. The transcript is defined as τ.

Input: pp, pw, sid, id (Alice), id' (Bob)

Setup: $pp \leftarrow \mathsf{HybridSetup}(1^\kappa)$ includes $(\mathbb{G}, g, q, \mathsf{KEM.params}, \mathcal{H})$

Initiator (Alice (i, id, pw)**)**	**Responder (Bob** (j, id', pw)**)**
Pass 1: Initial Exchange	
$x_1 \leftarrow \mathbb{Z}_q, x_2 \leftarrow \mathbb{Z}_q^*, ; X_1, X_2; \pi_{1,2}$	
$(PK_E^A, SK_E^A) \leftarrow \mathsf{KeyGen}$	
$\xrightarrow{X_1, X_2, \pi_{1,2}, \quad PK_E^A \quad , id}$	
Pass 2: Responder Response (Encrypted)	
	$x_3 \leftarrow \mathbb{Z}_q, x_4 \leftarrow \mathbb{Z}_q^*, ; X_3, X_4; \pi_{3,4}$
	$B = (X_1 X_2 X_3)^{x_4 \cdot pw}; \pi_B$
	$(ct_2, ss) \leftarrow \mathsf{Encaps}(PK_E^A)$
	$C_2 \leftarrow \mathsf{Enc}_{ss}(X_3, X_4, B, \pi_{3,4}, \pi_B)$
$\xleftarrow{ct_2, C_2 \quad , id'}$	
Pass 3: Initiator Final (Encrypted)	
$ss \leftarrow \mathsf{Decaps}(SK_E^A, ct_2)$	
$\mathsf{Dec}_{ss}(C_2) \rightarrow X_3, X_4, B, \pi_{3,4}, \pi_B$	
$A = (X_1 X_3 X_4)^{x_2 \cdot pw}; \pi_A$	
$C_3 \leftarrow \mathsf{Enc}_{ss}(A, \pi_A)$	
$\xrightarrow{C_3 \quad , id}$	
Key Computation	
	$\mathsf{Dec}_{ss}(C_3) \rightarrow A, \pi_A$
$K_1 = (B / g^{x_2 \cdot x_4 \cdot pw})^{x_2}$	$K_1 = (A / g^{x_2 \cdot x_4 \cdot pw})^{x_4}$
$K_2 = ss$	$K_2 = ss$

$$ssid := \tau \| id \| id'$$

$$K_{\mathsf{session}} = \mathsf{HKDF}(K_1 \| K_2 \| \tau \| \underbrace{id, id', pw}_{\text{context}})$$

Output $(K_{\mathsf{session}} \| id' \| ssid)$	Output $(K_{\mathsf{session}} \| id \| ssid)$

4.3 Parallel Composition of the Classical and PQ Layers

As mentioned earlier, our HPAKE protocol is formally constructed as a parallel composition of two independent components.

1. **Classical Key:** $K_1 = g^{(x_1+x_3)x_2 \cdot x_4 \cdot pw}$ (from J-PAKE).
2. **Post-Quantum Key:** $K_2 = ss$ (from ML-KEM).

The final key is the concatenation $K_{\mathsf{session}} = \mathsf{HKDF}(K_1 \| K_2 \| \tau)$. This structure aligns with the parallel composition framework of Lyu and Liu [34], whose Theorem 2 establishes that:

> *Parallel composition of multiple DH-type PAKEs yields a full DH-type PAKE (provably UC-secure) provided that (i) at least one component is full DH-type, and (ii) all other components are DH-type PAKEs that admit perfect simulation.*

We have verified these requirements in Section 3: J-PAKE satisfies condition (i), and the KEM layer satisfies condition (ii).

Although the J-PAKE flows are structurally nested within the KEM ciphertexts to provide immediate post-quantum protection, the cryptographic independence of the keys allows us to analyze the security as a parallel composition. The KEM simulator can replace the nested J-PAKE payloads with zeros without being detected (Lemma 8), effectively decoupling the components in the proof.

Entropy Mixing and Indistinguishability. The security of this composition relies on the properties of the Key Derivation Function (HKDF). Specifically, if one component's key is compromised, the other component's key must act as a source of fresh, indistinguishable entropy. Lemma 8 is critical here: it proves that the KEM key K_2 satisfies this indistinguishability requirement. Even if the classical key K_1 is fully known to the adversary (e.g., due to a quantum break of discrete log), K_2 remains pseudorandom. Since HKDF acts as a randomness extractor, the final hybrid key K_{session} remains indistinguishable from random as long as the KEM layer remains secure.

4.4 Security Analysis of the Hybrid PAKE

Theorem 2 (UC Security of the Hybrid PAKE). *Let Π_{HPAKE} be the hybrid protocol defined in Table 5. Assume the J-PAKE component $\Pi_{\mathsf{J-PAKE}}$ is a Full DH-type PAKE (under the* Gap-CDH *and* DDH *assumptions), the KEM component Π_{KEM} is a perfectly Simulatable DH-type protocol and is IND-CCA2 secure, and* HKDF *is a secure PRF, and the Random Oracle Model holds. Then, Π_{HPAKE} is a Full DH-type PAKE and UC-securely realizes $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$.*

Proof. Our proof leverages the Parallel Composition framework for DH-type PAKEs established by Lyu and Liu [34]. While our key derivation is parallel ($K = \mathsf{HKDF}(K_1 \| K_2 \| \tau)$), our transcript employs a *nested* structure where J-PAKE messages are encapsulated within KEM ciphertexts. Consequently, rather

than invoking the Parallel Composition Theorem (Theorem 2 of [34]) as a black box, we extend its reasoning to demonstrate that the Hybrid Protocol Π_{HPAKE} itself directly satisfies the definition of a Full DH-type PAKE as well, and thus the parallel composition is not violated. We define an explicit separation of adversarial capabilities across the protocol lifecycle: the UC simulation evaluates an *active classical network adversary* $\mathcal{A}$ manipulating communication flows during the real-time execution phase, whereas long-term forward secrecy evaluates a *passive quantum adversary* $\mathcal{A}_Q$ analyzing harvested transcripts post-execution. The intuition for this reduction is straightforward: We first rely on the IND-CCA2 security of the KEM to computationally "peel off" the outer ciphertext layer against the active real-time adversary. In the simulation, we replace the valid KEM ciphertexts (containing J-PAKE messages) with encryptions of random strings. This step decouples the adversary's view of the PAKE component from the KEM transcript entirely independently of the classical layer's specific structure. Once the layers are computationally decoupled (relying on the KEM's IND-CCA2 security for the reduction), the security of the hybrid protocol reduces to the properties of the underlying J-PAKE component. Since J-PAKE is a Full DH-type PAKE (Lemmas 1 to 5), the hybrid protocol inherits its *Computational Simulatability*, *One-Wayness*, and *Unique Password* properties. Thus, Π_{HPAKE} satisfies the Full DH-type definition against real-time active attacks and hence the UC security. We note that while the formal reduction utilizes the KEM's security to demonstrate structural independence, the One-Wayness of the session key remains protected by the classical J-PAKE component even if the post-quantum assumptions are eventually compromised (since $K_{\mathsf{session}} = \mathsf{HKDF}(K_1 \| K_2 \| \tau)$ remains secure as long as K_1 has high entropy). The detailed verification of these properties and the formal reduction appear in Appendix D.10.

Theorem 3 (Post-Quantum Forward Secrecy against HNDL). *The hybrid protocol Π_{HPAKE} satisfies Post-Quantum Forward Secrecy (Definition 4) against a passive quantum adversary $\mathcal{A}_Q$, provided the KEM satisfies IND-CCA2 security (under the MLWE assumption), the symmetric encryption scheme (SE) is IND-CPA secure against passive quantum adversaries, and HKDF is a secure PRF.*

Proof. We demonstrate that the final session key remains pseudorandom even if the adversary is given the correct password post-execution and later uses passive quantum power to solve the classical discrete logarithms. We rely on the IND-CCA2 security of the KEM to idealize the transcript and temporarily replace the KEM-derived secret (K_2) with a uniform random string. Subsequently, the security of HKDF as a PRF ensures that the final session key $K_{\mathsf{session}} = \mathsf{HKDF}(K_1 \parallel K_2 \parallel \tau)$ remains indistinguishable from random. We prove this via a sequence of games, Let S_i be the event that $\mathcal{A}_Q$ outputs 1 in Game_i.

Game 0 (Real World). The challenger runs the protocol honestly using password pw to yield $(\tau_{\mathsf{real}}, K_{\mathsf{real}})$. The adversary is given $(pw, \tau_{\mathsf{real}}, K_{\mathsf{real}})$. Hence, $\Pr[S_0] = \Pr[\mathcal{A}_Q(pw, \tau_{\mathsf{real}}, K_{\mathsf{real}}) = 1]$.

Game 1 (KEM Idealization). We modify the transcript τ and the intermediate KEM key K_2. Invoking Lemma 8, we replace the KEM ciphertexts C_2, C_3 with encryptions of zeros, and replace the KEM-derived shared secret with a uniform random string $K_2^* \leftarrow \$\mathcal{K}$. This yields τ_{sim} and a hybrid key $K_{\mathsf{session}} = \mathsf{HKDF}(K_1 \parallel K_2^* \parallel \tau_{\mathsf{sim}})$. By the IND-CCA2 security of the KEM: $|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}}(\mathcal{B}_1)$.

Game 2 (Key Randomness). In this game, we replace K_{session} with a completely random string $K_{\mathsf{rand}} \leftarrow \$\mathcal{K}$. Although the adversary $\mathcal{A}_Q$ knows pw post-execution and can theoretically use its passive quantum power to compute the classical component K_1 from the idealized transcript, the input key material to the HKDF is concatenated with K_2^*. Since K_2^* is a high-entropy uniform random string hidden entirely from the adversary (from Game 1), it acts as a secure cryptographic key for the PRF, while K_1 and τ_{sim} act as public inputs. The output of the PRF is computationally indistinguishable from K_{rand}. Hence: $|\Pr[S_1] - \Pr[S_2]| \leq \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}}(\mathcal{D})$.

Game 3 (Transcript Restoration). To match the challenge experiment where $b = 0$, we must restore the transcript back to the real execution. We restore the symmetric ciphertexts C_2, C_3 so they encrypt the true classical J-PAKE payloads A, B instead of zeros, yielding the real transcript τ_{real}. The view is now $(pw, \tau_{\mathsf{real}}, K_{\mathsf{rand}})$. Because K_2^* is a truly uniform random symmetric key independent of K_{rand}, distinguishing this restoration relies on the one-time semantic security of the symmetric encryption scheme. By the IND-CPA security of SE: $|\Pr[S_2] - \Pr[S_3]| \leq 2 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SE}}(\mathcal{B}_2)$. *(Note: The factor of 2 accounts for replacing the two distinct payload flows via a standard hybrid argument).*

Game 4 (Dummy Key World). The view in Game 3 is exactly $(pw, \tau_{\mathsf{real}}, K_{\mathsf{rand}})$, which is identical to the challenge distribution where the key is replaced with random ($b = 0$). Thus: $\Pr[S_3] = \Pr[S_4]$.

Combining the intermediate game transitions, the total advantage of $\mathcal{A}_Q$ in distinguishing the real session key from a random key is bounded by:

$$\mathsf{Adv}^{\mathsf{HNDL-SKS}}_{\Pi}(\mathcal{A}_Q) \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}}(\mathcal{B}_1) + \mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}}(\mathcal{D}) + 2 \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{SE}}(\mathcal{B}_2)$$

Provided the MLWE assumption holds, SE provides post-quantum symmetric indistinguishability, and HKDF is a secure PRF, this advantage is negligible.

Corollary 2 (Generic 3-Pass UC Compiler). *Let Π_{PAKE} be any 3-pass password-authenticated key exchange protocol that realizes the functionality $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$ against active adversaries under a set of classical assumptions. Let Π_{KEM} be a post-quantum KEM satisfying IND-CCA2 security. Then, the compiled protocol $\Pi_{\mathsf{Hyb}} = \mathsf{Compiler}_{\mathsf{HNDL}}(\Pi_{\mathsf{PAKE}}, \Pi_{\mathsf{KEM}})$, structured via the nested encapsulation wrapper defined in Section 4, UC-securely realizes the bare*

lazy-extraction functionality $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$. Consequently, applying standard session-identifier password transformations securely lifts this realization to the full multi-session functionalities $\mathcal{F}_{\mathsf{PAKE}^{\mathsf{LE}}}$, while achieving Post-Quantum Forward Secrecy against HNDL attacks.

Proof. The proof follows from Theorem 3 and the UC security of the underlying PAKE. Because the KEM wrapper acts as an active, content-agnostic transport layer during real-time execution, the IND-CCA2 reduction idealizes the ciphertexts entirely independently of the underlying PAKE's specific structure. Once idealized, the simulated view reduces directly to the native active UC security guarantees of Π_{PAKE} and the hybrid protocol preserves its functionality, while the independent, high-entropy KEM secret guarantees long-term HNDL resilience via the PRF security of the final HKDF.

Theorem 4 (Passive UC Security via Pure Post-Quantum Assumptions). *Consider a passive adversary $\mathcal{A}_{\mathsf{pass}}$ that observes the execution transcripts of the hybrid protocol Π_{HPAKE} but does not inject or modify messages. If the post-quantum KEM satisfies IND-CPA security (implied by the MLWE assumption) and HKDF is a secure PRF, then Π_{HPAKE} achieves full passive session-key secrecy without relying on any classical assumptions (*$\mathsf{Gap\text{-}CDH}$ *or* DDH*).*

Proof. In a purely passive scenario, the honest parties Alice and Bob execute the protocol faithfully, generating valid, matching ephemeral keypairs and deriving an identical, unmanipulated KEM shared secret $K_2 = ss$. Because $\mathcal{A}_{\mathsf{pass}}$ is passive, no active extraction or password-guessing attacks occur during the execution phase. Consequently, we can directly invoke the IND-CPA security of the KEM. We define a game hop where the true encapsulated secret ss is replaced with a uniformly random string $ss^* \leftarrow \$\mathcal{K}$. Under the MLWE assumption, the adversary's advantage in detecting this replacement is bounded by $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM}}(\mathcal{A}_{\mathsf{pass}})$. Once the KEM secret is idealized to a uniform random string ss^*, the final session key is derived as $K_{\mathsf{session}} = \mathsf{HKDF}(K_1 \parallel ss^* \parallel \tau)$. Because ss^* is uniformly random and hidden from $\mathcal{A}_{\mathsf{pass}}$, it acts as a secure, high-entropy key for the PRF. Thus, the output K_{session} is computationally indistinguishable from a truly random string, governed entirely by $\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{HKDF}}$. Crucially, this reduction does not evaluate, extract, or rely upon the classical shared secret K_1. Even if K_1 is entirely compromised or provides zero entropy (e.g., assuming an absolute failure of $\mathsf{Gap\text{-}CDH}$ and DDH), the uniformity of ss^* guarantees the pseudorandomness of the final session key. Therefore, passive security is achieved relying exclusively on post-quantum assumptions.

While HNDL security explicitly models a passive harvesting phase, we also address an active adversary $\mathcal{A}$ who manipulates the KEM layer during execution to bypass the post-quantum encryption of the J-PAKE transcript.

Proposition 1 (Resilience to Active KEM Interference). *We consider an active adversary $\mathcal{A}$ who modifies the ephemeral KEM public keys ($PK \rightarrow PK'$) during execution to bypass post-quantum encryption. In such a scenario, the*

honest parties P_I and P_R derive divergent session keys ($K_I \neq K_R$) because the public keys are cryptographically bound into the HKDF *inputs. Consequently, the explicit key confirmation step (Pass 3) fails verification, causing the honest parties to abort the session with overwhelming probability. Regarding HNDL-Session Key Secrecy, since the session is aborted, no valid session key is ever established, and the adversary gains no information about any honest keys. While $\mathcal{A}$ may successfully strip the KEM layer and potentially recover pw from the decrypted J-PAKE values using a future quantum computer, this does not violate the HNDL security, as it explicitly grants the adversary knowledge of pw to test the security of the session key. Therefore, active manipulation of the KEM layer results in a Denial-of-Service but yields no advantage against the confidentiality of established session keys.*

Our hybrid PAKE design is intended to support the post-quantum transition rather than to provide full post-quantum security, as we do not consider an active quantum adversary. Instead, our security model focuses on HNDL attacks, ensuring that past session keys remain secure against a future quantum adversary, even if the password is later compromised. We leave the study of fully post-quantum PAKE designs against active adversaries with access to a quantum computer to future work.

5 Practical Evaluation

To demonstrate the practical viability and computational efficiency of our hybrid protocol, we built a reference implementation in Java using the `BouncyCastle` library (v1.78+). The implementation employs the NIST standard curve P-256 (`secp256r1`) for classical PAKE (J-PAKE) ephemerals, ML-KEM-768 for post-quantum key encapsulation, AES-256-GCM for AEAD cipher payload wrapping, and SHA-256 for HKDF key derivation and Fiat-Shamir NIZK proof challenges. The benchmarks were executed on a MacBook Air laptop equipped with an Apple M4 processor and 16 GB of memory, running macOS Tahoe 26.3.1 and Java OpenJDK 24.0.2. To account for JVM Just-In-Time (JIT) compilation optimizations, execution times were averaged over 1000 measured iterations following 100 warm-up cycles.

Table 6 summarizes the performance metrics across each phase of the protocol. Overall, the hybrid construction is highly efficient, completing a full 3-pass handshake in just 2.81 ms with a total communication payload of 2,916 bytes. The Initiator requires a total of 1.43 ms of computation time (0.37 ms in Pass 1 and 1.06 ms in Pass 3) while transmitting 1,506 bytes, and the Responder requires 1.37 ms of computation time (0.95 ms in Pass 2 and 0.42 ms in Pass 3) while transmitting 1,410 bytes. Notably, the post-quantum lattice operations and symmetric encryption contribute negligibly to the overall processing overhead, which remains dominated by classical elliptic curve scalar multiplications. These results confirm that our hybrid wrapper achieves post-quantum confidentiality without introducing meaningful latency or bandwidth bottlenecks.

Table 6: Computation and Communication Breakdown of the Hybrid Protocol. Communication payload sizes assume compressed elliptic curve points (33 bytes) over `secp256r1`, scalar values in $\mathbb{Z}_q$ (32 bytes), Schnorr NIZK proofs (65 bytes), ML-KEM-768 (PK_E^A = 1184 bytes, ct_2 = 1088 bytes), and AES-256-GCM (28 bytes overhead, comprising a 12-byte nonce and a 16-byte MAC tag).

Pass	Party	Operations	Comm. (Bytes)	Comp. (ms)
Pass 1	Initiator	$(PK_E^A, SK_E^A) \leftarrow \mathsf{KeyGen}()$	–	0.03
		Gen X_1, X_2, π_1, π_2	–	0.35
		Send: $(X_1, X_2, \pi_1, \pi_2, PK_E^A)$	1380	–
		Pass 1 Subtotal	**1380**	**0.37**
Pass 2	Responder	Verify π_1, π_2	–	0.37
		$(ct_2, ss) \leftarrow \mathsf{Encaps}(PK_E^A)$	–	0.03
		Gen $X_3, X_4, B, \pi_3, \pi_4, \pi_B$	–	0.55
		$C_2 \leftarrow \mathsf{Enc}_{ss}(X_3, X_4, B, \pi_{3,4}, \pi_B)$	–	0.01
		Send: (ct_2, C_2)	1410	–
		Pass 2 Subtotal	**1410**	**0.95**
Pass 3	Initiator	$ss \leftarrow \mathsf{Decaps}(SK_E^A, ct_2)$	–	0.03
		$\mathsf{Dec}_{ss}(C_2)$	–	0.01
		Verify π_3, π_4, π_B	–	0.57
		Gen A, π_A, $C_3 \leftarrow \mathsf{Enc}_{ss}(A, \pi_A)$	–	0.21
		Send: C_3	126	–
		Compute K_1, Derive K_{session}	–	0.19
		Pass 3 Subtotal	**126**	**1.06**
	Responder	$\mathsf{Dec}_{ss}(C_3)$	–	0.005
		Verify π_A	–	0.20
		Compute K_1, Derive K_{session}	–	0.19
		Pass 3 Subtotal	**–**	**0.42**
Total			**2916**	**2.81**

6 Conclusion

In this paper, we proposed a novel hybrid PAKE solution to prevent the harvest-now-decrypt-later (HNDL) attacks against a prospective quantum adversary. Instead of combining two PAKE protocols and relying on an abstract post-quantum PAKE scheme that is yet to achieve maturity, our approach was to leverage concrete, mature techniques by integrating an established classic PAKE protocol into a standardized KEM layer. As a concrete instantiation, we presented a hybrid J-PAKE scheme, which preserves the established security of J-PAKE in the classic setting, whilst providing the enhanced security against HNDL attacks. We formally proved the security of the proposed protocol in a UC model. Finally, we provided an open-source implementation and performance evaluation.

In our evaluation, the complete key exchange process took only 2.81 ms in the computation. This shows the practical feasibility of our solution.

Source Code Availability

The implementation of the Hybrid PAKE (J-PAKE + Kyber) is anonymized and available at: https://osf.io/xa3kp/overview?view_only=91165cff338c42408f785d8a5cd5f91c.

References

1. Abdalla, M., Barbosa, M., Bradley, T., Jarecki, S., Katz, J., Xu, J.: Universally composable relaxed password authenticated key exchange. In: Advances in Cryptology – CRYPTO 2020. pp. 278–307. Springer (2020)
2. Abdalla, M., Benhamouda, F., MacKenzie, P.: Security of the j-pake password-authenticated key exchange protocol. In: 2015 IEEE Symposium on Security and Privacy. pp. 571–587 (2015)
3. Abdalla, M., Benhamouda, F., Pointcheval, D.: Public-key encryption indistinguishable under plaintext-checkable attacks. In: Public-Key Cryptography – PKC 2015. pp. 332–352. Springer (2015)
4. Abdalla, M., Eisenhofer, T., Kiltz, E., Kunzweiler, S., Riepel, D.: Password-authenticated key exchange from group actions. In: Annual international cryptology conference. pp. 699–728. Springer (2022)
5. Abdalla, M., Haase, B., Hesse, J.: Security analysis of cpace. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 711–741. Springer (2021)
6. Abdalla, M., Pointcheval, D.: Simple password-based encrypted key exchange protocols. In: Topics in Cryptology – CT-RSA 2005. pp. 191–208. Springer (2005)
7. Alnahawi, N., Haas, D., Mauß, E., Wiesmaier, A.: SoK: PQC PAKEs - design, security and performance. Cryptology ePrint Archive, Paper 2025/119 (2025)
8. Arriaga, A., Barbosa, M., Jarecki, S., Škrobot, M.: C'est très CHIC: A compact password-authenticated key exchange from lattice-based KEM. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 3–33. Springer (2024)
9. Barbosa, M., Gellert, K., Hesse, J., Jarecki, S.: Bare pake: Universally composable key exchange from just passwords. In: Advances in Cryptology – CRYPTO 2024. pp. 183–217. Springer (2024)
10. Beguinet, H., Chevalier, C., Pointcheval, D., Ricosset, T., Rossi, M.: Get a cake: Generic transformations from key encapsulation mechanisms to password authenticated key exchanges. In: International Conference on Applied Cryptography and Network Security. pp. 516–538. Springer (2023)
11. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Advances in Cryptology — EUROCRYPT 2000. pp. 139–155. Springer (2000)
12. Bellovin, S.M., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy pp. 72–84 (1992)

13. Boneh, D.: The decision diffie-hellman problem. In: Algorithmic Number Theory. pp. 48–63. Springer (1998)
14. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehle, D.: CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367 (2018)
15. Boyko, V., MacKenzie, P., Patel, S.: Provably secure password-authenticated key exchange using diffie-hellman. In: Advances in Cryptology — EUROCRYPT 2000. pp. 156–171. Springer (2000)
16. Canetti, R., Dachman-Soled, D., Vaikuntanathan, V., Wee, H.: Efficient password authenticated key exchange via oblivious transfer. In: Public Key Cryptography – PKC 2012. pp. 449–466. Springer (2012)
17. Canetti, R., Rabin, T.: Universal composition with joint state. In: Advances in Cryptology - CRYPTO 2003. pp. 265–281. Springer (2003)
18. Ding, J., Alsayigh, S., Lancrenon, J., Saraswathy, R.V., Snook, M.: Provably secure password authenticated key exchange based on rlwe for the post-quantum world. In: The Cryptographer's Track at RSA Conference (2017)
19. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 524–543. Springer (2003)
20. Groce, A., Katz, J.: A new framework for efficient password-based authenticated key exchange. In: Proceedings of the 17th ACM conference on Computer and communications security. pp. 516–525 (2010)
21. Groth, J.: Simulation-sound nizk proofs for a practical language and constant size group signatures. In: Advances in Cryptology – ASIACRYPT 2006. pp. 444–459. Springer (2006)
22. Haase, B., Labrique, B.: Aucpace: Efficient verifier-based pake protocol tailored for the iiot. IACR Trans. Cryptogr. Hardw. Embed. Syst. **2019**, 1–48 (2019)
23. Hao, F., van Oorschot, P.C.: SoK: password-authenticated key exchange–theory, practice, standardization and real-world lessons. In: Proceedings of the 2022 ACM on Asia conference on computer and communications security. pp. 697–711 (2022)
24. Hao, F., Ryan, P.Y.A.: J-PAKE: Authenticated key exchange without PKI. In: Transactions on Computational Science XI, Lecture Notes in Computer Science, vol. 6480, pp. 192–206. Springer (2010)
25. Harkins, D.: Simultaneous authentication of equals: A secure, password-based key exchange for mesh networks. 2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008) pp. 839–844 (2008)
26. Hesse, J., Rosenberg, M.: Pake combiners and efficient post-quantum instantiations. In: Advances in Cryptology – EUROCRYPT 2025. pp. 395–420. Springer (2025)
27. Ishibashi, R., Yoneyama, K.: Compact password authenticated key exchange from group actions. In: Information Security and Privacy. pp. 220–247. Springer (2023)
28. Jablon, D.P.: Strong password-only authenticated key exchange. Comput. Commun. Rev. **26**, 5–26 (1996)
29. Katz, J., Vaikuntanathan, V.: Smooth projective hashing and password-based authenticated key exchange from lattices. In: Advances in Cryptology – ASIACRYPT 2009. pp. 636–652. Springer (2009)
30. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Theory of Cryptography. pp. 293–310. Springer (2011)
31. Krawczyk, H.: Cryptographic extraction and key derivation: The hkdf scheme. In: Annual Cryptology Conference. pp. 631–648. Springer (2010)

32. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography **75**(3), 565–599 (2015)
33. Liu, X., Liu, S., Han, S., Gu, D.: Eke meets tight security in the universally composable framework. In: IACR International Conference on Public-Key Cryptography. pp. 685–713. Springer (2023)
34. Lyu, Y., Liu, S.: Hybrid password authentication key exchange in the uc framework. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 421–450. Springer (2025)
35. Lyu, Y., Liu, S., Han, S.: Universal composable password authenticated key exchange for the post-quantum world. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 120–150. Springer (2024)
36. MacKenzie, P.: On the security of the speke password-authenticated key exchange protocol. Cryptology ePrint Archive (2001)
37. Okamoto, T., Pointcheval, D.: The gap-problems: A new class of problems for the security of cryptographic schemes. In: Public Key Cryptography. pp. 104–118. Springer (2001)
38. Pointcheval, D., Wang, G.: VTBPEKE: Verifier-based Two-Basis Password Exponential Key Exchange. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. p. 301–312 (2017)
39. Santos, B.F.D., Gu, Y., Jarecki, S.: Randomized half-ideal cipher on groups with applications to uc (a) pake. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 128–156. Springer (2023)
40. Shoup, V.: Security analysis of spake2. In: Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, November 16–19, 2020, Proceedings, Part III. vol. 12552, p. 31. Springer Nature (2020)
41. Steiner, M., Tsudik, G., Waidner, M.: Diffie-hellman key distribution extended to group communication. In: Conference on Computer and Communications Security (1996)
42. Vos, J., Jarecki, S., Wood, C.A., Yun, C., Myers, S., Sierra, Y.: A hybrid asymmetric password-authenticated key exchange in the random oracle model. Cryptology ePrint Archive (2025)

A Formal Definition of the Ideal Functionality

Ideal Functionality $\mathcal{F}_{\mathsf{bPAKE}}^{\mathsf{LE}}$

Parameters: Key space $\mathcal{K}$ and a set of used sub-session identifiers $\mathcal{S}$ (initialized to $\emptyset$).
State: The functionality maintains records for registered users, and active/completed sessions ($\mathsf{ses}_{\mathsf{act}}$, $\mathsf{ses}_{\mathsf{hbc}}$, $\mathsf{late}_{\mathsf{test}}$).

1. Session Initiation
Input: $(\mathsf{NewSession}, sid, i, pw, id, role)$ from party P.

- **Notify:** Send $(\mathsf{NewSession}, sid, P, i, id, role)$ to adversary $\mathcal{A}$.
- **Record:** If no record exists for (P, i), store (P, i, pw, id).

2. Key Generation (Active)
Input: $(\mathsf{ActiveNewKey}, sid, P, i, pw', K', ssid, cpid)$ from $\mathcal{A}$.

- **Precondition:** Record $(P, i, [pw, id])$ must exist.
- **Replay Check:** If record $(\mathsf{ses}_{\mathsf{act}}, P, i, ssid, cpid, [K])$ exists, output $(sid, i, K, ssid, cpid)$ to P and halt.
- **Processing:** If $ssid \notin \mathcal{S}$:
  1. $\mathcal{S} \leftarrow \mathcal{S} \cup \{ssid\}$.
  2. **Key Determination:** If $pw' = pw$, set $K := K'$. Else, sample $Key \xleftarrow{\$} \mathcal{K}$.
  3. **Lazy Hook:** If $pw' = \perp$, sample $Key \xleftarrow{\$} \mathcal{K}$ and save $(\mathsf{late}_{\mathsf{test}}, P, ssid, pw, K)$.
  4. **Record:** Save $(\mathsf{ses}_{\mathsf{act}}, P, i, ssid, cpid, K)$.
  5. **Output:** Send $(sid, i, K, ssid, cpid)$ to P.

3. Key Generation (Passive)
Input: $(\mathsf{PassiveNewKey}, sid, P, i, CP, j, ssid)$ from $\mathcal{A}$.

- **Precondition:** Record $(P, i, [pw, id])$ must exist.
- **Replay Check:** If record $(\mathsf{ses}_{\mathsf{hbc}}, P, i, CP, j, ssid, [cpid, K])$ exists, output $(sid, i, K, ssid, cpid)$ to P and halt.
- **Derive Peer:** If no record $(CP, j, [pw', id'])$ exists, ignore and halt.
  Set $cpid := id'$.
- **Processing:**
  - **Case 1: First-to-Terminate** ($ssid \notin \mathcal{S}$)
    1. $\mathcal{S} \leftarrow \mathcal{S} \cup \{ssid\}$.
    2. Sample $Key \xleftarrow{\$} \mathcal{K}$.
    3. Save $(\mathsf{ses}_{\mathsf{hbc}}, P, i, CP, j, ssid, cpid, K)$.
    4. Output $(sid, i, K, ssid, cpid)$ to P.
  - **Case 2: Second-to-Terminate** (Record $(\mathsf{ses}_{\mathsf{hbc}}, CP, j, P, i, ssid, id, [K'])$ exists)
    1. **Key Agreement:** If $pw = pw'$, set $K := K'$. Else, sample $Key \xleftarrow{\$} \mathcal{K}$.
    2. Save $(\mathsf{ses}_{\mathsf{hbc}}, P, i, \perp, \perp, ssid, cpid, K)$.
    3. Output $(sid, i, K, ssid, cpid)$ to P.

4. Late Password Test (Offline Attack)
Input: $(\mathsf{LateTestPwd}, sid, P, i, ssid, pw')$ from $\mathcal{A}$.

- **Check:** If record $(\mathsf{late}_{\mathsf{test}}, P, ssid, [pw, K])$ exists:
  1. Delete the record.
  2. If $pw' = pw$, return K to $\mathcal{A}$.
  3. Else, return $Key' \xleftarrow{\$} \mathcal{K}$ to $\mathcal{A}$.

Fig. 3: The Bare PAKE Ideal Functionality $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$, adapted from Barbosa et al. [9] and Lyu-Liu [34].

B Assumptions & Definitions

Classical Assumptions. Let $\mathbb{G}$ be a cyclic group of prime order q generated by g. The security of the classical J-PAKE component [24] relies on the hardness of the *Decisional Diffie-Hellman (Assumption 5)* and *Gap Computational Diffie-Hellman (Assumption 6)* problems in $\mathbb{G}$ [37,13]. Additionally, we require a *Simulation-Sound Extractable Non-Interactive Zero-Knowledge* (SE-NIZK) proof system for discrete logarithms (Assumption 7). To achieve concurrent UC security without rewinding, this system must provide *straight-line extraction*. Following Abdalla et al. [2], this is achieved in the Random Oracle Model (ROM) by restricting the analysis to algebraic adversaries, where an extractor can recover the witness with overwhelming probability without rewinding the adversary.

Assumption 5 (Decisional Diffie-Hellman (DDH) [13]) *Let $\mathbb{G}$ be a cyclic group of prime order q generated by g. An algorithm for the* Decision Diffie-Hellman *problem is given $(X, Y, Z) \in \mathbb{G}^3$ and must decide whether $Z = g^{xy}$ or Z is a uniformly random element of $\mathbb{G}$. For a PPT adversary $\mathcal{A}$, define its* DDH *advantage as* $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{DDH}}(\mathcal{A}) = \Big|\Pr\big[x, y \xleftarrow{\$} \mathbb{Z}_q;\ X \leftarrow g^x,\ Y \leftarrow g^y,\ Z \leftarrow g^{xy} : \mathcal{A}(X,Y,Z) = 1\big] - \Pr\big[x, y, z \xleftarrow{\$} \mathbb{Z}_q;\ X \leftarrow g^x,\ Y \leftarrow g^y,\ Z \leftarrow g^z : \mathcal{A}(X,Y,Z) = 1\big]\Big|$. *We say that the* DDH *assumption holds in $\mathbb{G}$ if $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{DDH}}(\mathcal{A})$ is negligible for all PPT adversaries $\mathcal{A}$.*

Assumption 6 (Computational Diffie-Hellman (CDH) and Gap-CDH [37]) *Let $\mathbb{G}$ be a cyclic group of prime order q generated by g. Let $x, y \xleftarrow{\$} \mathbb{Z}_q$, and set $X = g^x, Y = g^y$. The* Computational Diffie-Hellman (CDH) *problem is to compute $Z = g^{xy}$ given (X, Y). The* Gap Computational Diffie-Hellman (Gap-CDH) *problem considers the same experiment but grants the adversary access to a Decisional Diffie–Hellman oracle $\mathcal{O}_{\mathsf{DDH}}$, defined as:*

$$\mathcal{O}_{\mathsf{DDH}}(X, Y, Z) = \begin{cases} 1 & \text{if } Z = g^{xy} \text{ where } X = g^x,\ Y = g^y, \\ 0 & \text{otherwise.} \end{cases}$$

The Gap-CDH *assumption holds in $\mathbb{G}$ if for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$, $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{Gap\text{-}CDH}}(\mathcal{A}) = \Pr\left[\mathcal{A}^{\mathcal{O}_{\mathsf{DDH}}}(g, g^x, g^y) = g^{xy}\right] \leq \mathsf{negl}(\kappa)$.*

Assumption 7 (Simulation-Sound Extractable NIZK [21]) *Let $\mathcal{L}_{\mathsf{DL}} = \{(u, h) \in \mathbb{G}^2 \mid \exists r \in \mathbb{Z}_q : u = h^r\}$ be the language of discrete logarithm tuples. We assume the existence of a* labeled Simulation-Sound Extractable Non-Interactive Zero-Knowledge *(SE-NIZK) proof system $\Pi_{\mathsf{NIZK}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Sim}, \mathsf{Ext})$ satisfying:*

1. ***Completeness:** For any valid statement and witness, the honest prover always produces an accepting proof.*

2. ***Zero-Knowledge:*** *Simulated proofs (generated by* Sim *without a witness) are computationally indistinguishable from real proofs.*
3. ***Simulation-Sound Extractability:*** *For any PPT adversary* $\mathcal{A}$ *with access to the simulation oracle* Sim*, if* $\mathcal{A}$ *outputs a valid proof* π *for a statement* (u, h) *(and* π *was not generated by* Sim*), then the extractor* Ext *extracts the witness* r *such that* $u = h^r$ *with overwhelming probability.*
4. ***Straight-Line Extraction:*** *The extractor* Ext *runs in strict polynomial time and* ***does not rewind*** *the adversary.*

Remark 2. This assumption is critical for the J-PAKE security analysis. It ensures that for any valid proof submitted by an adversary, the discrete logarithm witness is well-defined and extractable. This prevents attacks where an attacker could construct a single protocol message that appears valid relative to multiple different passwords. Crucially, the straight-line extraction requirement avoids rewinding, which is necessary for concurrent UC security. As noted earlier, while standard Schnorr proofs typically require rewinding (via the Forking Lemma) in the Random Oracle Model, Abdalla et al. [2] have shown that they admit straight-line extraction in the Random Oracle Model by restricting attention to algebraic adversaries. In this model, the extractor can recover the witness directly from the adversary's algebraic representation of the group elements, enabling extraction without rewinding. This weaker but sufficient notion allows our reduction to remain compatible with the UC framework.

Post-Quantum Assumptions. Post-quantum confidentiality is established via an Key Encapsulation Mechanism (e.g., Kyber) [14]. The security of Kyber relies on the *Module-Learning With Error (MLWE)* problem [32] (Assumption 8), which implies the IND-CCA2 security of the KEM in the ROM via the Fujisaki-Okamoto transform.

Assumption 8 (Module-LWE (MLWE) [32]) *Let* n, k, q *be integer parameters and* η *be an error distribution parameter as defined in Kyber [14]. The Module-LWE problem over the ring* $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ *asks an adversary to distinguish between samples from the uniform distribution over* $R_q^k \times R_q$ *and samples from the distribution* $(\mathbf{A}, \mathbf{t} = \mathbf{A}^T\mathbf{s} + \mathbf{e})$*, where* $\mathbf{A} \leftarrow R_q^{k \times k}$*, and secrets/errors* $\mathbf{s}, \mathbf{e}$ *are sampled from the error distribution. We assume that for the parameters chosen, the advantage of any PPT adversary in distinguishing these distributions is negligible:* $\mathsf{Adv}_{k,\eta}^{\mathsf{MLWE}}(\mathcal{A}) \leq \mathsf{negl}(\kappa)$.

Implication for Kyber. The Kyber KEM is constructed by applying the Fujisaki–Okamoto (FO) transform to an IND-CPA secure encryption scheme based on MLWE. Consequently, the hardness of MLWE implies the IND-CCA2 security of the Kyber KEM in the Random Oracle Model [14].

Symmetric Encryption & Key Derivation. Protocol flows are symmetrically encrypted using an *Authenticated Encryption with Associated Data*

(AEAD) scheme (e.g., AES-GCM or ChaCha20-Poly1305), Definition 11. Because ephemeral keys are used for a single session, standard *Indistinguishability under Chosen-Plaintext Attack* (IND-CPA) and *Ciphertext Integrity* (INT-CTXT) security notions are sufficient (Definitions 12 and 13, respectively). Finally, the shared secrets K_1 (from J-PAKE) and K_2 (from KEM) are concatenated to form the input keying material (IKM), and the transcript serves as the salt/context. We use HKDF (RFC 5869) as a key derivation function. The security of the final key derivation relies on the assumption that HKDF acts as a secure *Pseudorandom Function (PRF)* [31] (Definition 15) (or a Random Oracle in the security proof) when keyed with high-entropy material.

Definition 11 (Symmetric Encryption Scheme). *A symmetric encryption scheme* $\Pi_{\mathsf{SE}} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *consists of three polynomial-time algorithms:*

- $\mathsf{KeyGen}(1^\kappa)$*: Outputs a secret key* $k \in \{0,1\}^\kappa$*.*
- $\mathsf{Enc}_k(m, \mathsf{aad})$*: Takes a key* k*, a plaintext* $m \in \{0,1\}^*$*, and associated data* $\mathsf{aad} \in \{0,1\}^*$*, and outputs a ciphertext* c*.*
- $\mathsf{Dec}_k(c, \mathsf{aad})$*: Takes a key* k*, a ciphertext* c*, and associated data* aad*, and outputs a plaintext* m *or a rejection symbol* $\perp$*.*

Correctness. For all $k \leftarrow \mathsf{KeyGen}(1^\kappa)$, all messages m, and all associated data aad we have: $\mathsf{Dec}_k(\mathsf{Enc}_k(m, \mathsf{aad}), \mathsf{aad}) = m$.

Security. We require the scheme to satisfy two properties: *Indistinguishability under Chosen-Plaintext Attack* (IND-CPA) and *Integrity of Ciphertexts* (INT-CTXT) (Definitions 12 and 13). These properties ensure that encrypted protocol messages do not leak information about the plaintext and that any modification of ciphertexts by an adversary is detected with overwhelming probability.

Definition 12 (IND-CPA Security). *The IND-CPA advantage of an adversary* $\mathcal{A}$ *against* Π_{SE} *is defined as* $\mathsf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE}}(\mathcal{A}) = \left|\Pr[b' = b] - \frac{1}{2}\right|$*, where the experiment is defined as: 1.* $k \leftarrow \mathsf{KeyGen}(1^\kappa)$*. 2.* $\mathcal{A}$ *outputs two messages* m_0, m_1 *of equal length and associated data* aad*. 3. The challenger samples* $b \leftarrow \{0,1\}$ *and computes* $c^* \leftarrow \mathsf{Enc}_k(m_b, \mathsf{aad})$*. 4.* $\mathcal{A}$ *receives* c^* *and outputs a bit* b'*.*

Note that as the keys in our protocol are ephemeral and used only once, this single-challenge definition suffices.

Definition 13 (Ciphertext Integrity (INT-CTXT)). *The INT-CTXT advantage of an adversary* $\mathcal{A}$ *is defined as the probability that* $\mathcal{A}$ *produces a valid forgery* $\mathsf{Adv}^{\mathsf{int\text{-}ctxt}}_{\mathsf{SE}}(\mathcal{A}) = \Pr\left[\mathsf{Dec}_k(c, \mathsf{aad}) \neq \perp \ \wedge \ (c, \mathsf{aad}) \notin \mathcal{Q}\right]$*, where* $k \leftarrow \mathsf{KeyGen}(1^\kappa)$*, and* $\mathcal{Q}$ *denotes the set of ciphertexts output by the encryption oracle queried by* $\mathcal{A}$*.*

Definition 14 (IND-CCA2 Security). *The IND-CCA2 advantage of an adversary $\mathcal{A}$ against a KEM is defined as:*

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca2}}_{\mathsf{KEM}}(\mathcal{A}) \;=\; \left| 2 \cdot \Pr \left[\begin{array}{l} (pk, sk) \leftarrow \mathsf{KeyGen}(1^\kappa); \\ b \leftarrow \{0,1\}, K_0^* \leftarrow \mathcal{K}; \\ (c^*, K_1^*) \leftarrow \mathsf{Encaps}(pk); \\ b' \leftarrow \mathcal{A}^{\mathsf{Decaps}(\cdot)}(pk, c^*, K_b^*) \end{array} \right] \;-\; 1 \right|,$$

where the oracle $\mathsf{Decaps}(\cdot)$ *returns* $\mathsf{Decaps}(sk, \cdot)$ *but returns* $\perp$ *if queried on the challenge ciphertext* c^*.

Definition 15 (Pseudorandom Function (PRF) [31]). *Let* $F : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^n$ *be a deterministic efficient function, where the first input is the key and the second is the input string. We say* F *is a* secure PRF *if for all probabilistic polynomial-time (PPT) distinguishers* $\mathcal{D}$*, the advantage* $\mathsf{Adv}^{\mathsf{PRF}}_F(\mathcal{D}) = \left|\Pr_{k \leftarrow \{0,1\}^\kappa}[\mathcal{D}^{F(k,\cdot)} = 1] - \Pr_{R \leftarrow \mathsf{Func}}[\mathcal{D}^{R(\cdot)} = 1]\right|$ *is negligible in* κ*, where* Func *denotes the set of all random functions with the same domain and range as* F.

C DH Mapping Tables

Table 7: Core computational mappings of the J-PAKE protocol phase

Algorithm	**Description**
$\mathsf{PreInit}(.)$	Sample $X_1 = g^{x_1}$, $X_2 = g^{x_2}$ and corresponding ZKPs (π_1 and π_2).
$\mathsf{PreResp}(.)$	Sample $X_3 = g^{x_3}$, $X_4 = g^{x_4}$ and corresponding ZKPs (π_3 and π_4).
$\mathsf{DHInit}(\cdot)$	Compute $A = (X_1 X_3 X_4)^{x_2 \cdot pw}$ and corresponding ZKPs (π_A).
$\mathsf{DHResp}(\cdot)$	Compute $B = (X_1 X_2 X_3)^{x_4 \cdot pw}$ and corresponding ZKPs (π_B).
$\mathsf{Compi}(\cdot)$	Compute $K = (\frac{B}{X_2^{x_4 \cdot pw}})^{x_2} = (X_1 X_3)^{x_2 \cdot x_4 \cdot pw}$.
$\mathsf{Compr}(\cdot)$	Compute $K = (\frac{A}{X_4^{x_2 \cdot pw}})^{x_4} = (X_1 X_3)^{x_2 \cdot x_4 \cdot pw}$.

Table 8: Simulation algorithms for 3-pass J-PAKE. Trapdoor td denotes the extraction key of the SE-NIZK system and is used exclusively to extract witnesses from accepting adversarial proofs.

Algorithm	**Description**
$\mathsf{SimPreInit}(pp)$	Sample $x_1, x_2 \leftarrow \mathbb{Z}_q$. Set $X_1 := g^{x_1}, X_2 := g^{x_2}$. Simulate proofs π'_1, π'_2. Return $m_0^{(I)} = (X_1, X_2, \pi'_1, \pi'_2)$ and $st_0^{(I)} = (x_1, x_2)$.
$\mathsf{SimPreResp}(pp)$	Sample $x_3, x_4 \leftarrow \mathbb{Z}_q$. Set $X_3 := g^{x_3}, X_4 := g^{x_4}$. Simulate proofs π'_3, π'_4. Return $m_0^{(R)} = (X_3, X_4, \pi'_3, \pi'_4)$ and $st_0^{(R)} = (x_3, x_4)$.
$\mathsf{SimDHResp}(pp,$ $st_0^{(R)}, m_0^{(I)})$	Parse $st_0^{(R)}$ and $m_0^{(I)}$. Sample random $r' \leftarrow \mathbb{Z}_q$. Compute $B := (X_1X_2X_3)^{r'}$ (or $B \leftarrow \mathbb{G}$). Simulate proof π'_B. Return $m_2 = (B, \pi'_B)$ and $st_1^{(R)} = x_4$.
$\mathsf{SimDHInit}(pp,$ $st_0^{(I)}, m_{0,2}^{(R)})$	Parse $st_0^{(I)}$ and $m_0^{(R)}$. Sample random $r' \leftarrow \mathbb{Z}_q$. Compute $A := (X_1X_3X_4)^{r'}$ (or $A \leftarrow \mathbb{G}$). Simulate proof π'_A. Return $m_1 = (A, \pi'_A)$ and $st_1^{(I)} = x_2$.
$\mathsf{SimCompi}(td, pw,$ $st_1^{(I)}, \ldots, m_2)$	Parse $st_1^{(I)} = x_2$. Extract witness $x_4 \leftarrow \mathsf{Ext}(td, \pi_4)$. Compute key using extracted witness: $K := (X_1X_3)^{x_2 \cdot x_4 \cdot pw}$.
$\mathsf{SimCompr}(td, pw,$ $st_1^{(R)}, \ldots, m_1)$	Parse $st_1^{(R)} = x_4$. Extract witness $x_2 \leftarrow \mathsf{Ext}(td, \pi_2)$. Compute key using extracted witness: $K := (X_1X_3)^{x_2 \cdot x_4 \cdot pw}$.

Table 9: Algorithmic definition of the KEM component as a 3-pass DH-type protocol.

Algorithm	**Description**
$\mathsf{DHSetup}^{\mathsf{KEM}}(1^\kappa)$	Generate global KEM parameters. Output $(pp^{\mathsf{KEM}}, \perp)$.
$\mathsf{PreInit}^{\mathsf{KEM}}(\cdot)$	Generate ephemeral keypair $(PK_E^A, SK_E^A) \leftarrow \mathsf{KeyGen}$. Return $m_0^{(I)} = PK_E^A$ and $st_0^{(I)} = SK_E^A$.
$\mathsf{DHResp}^{\mathsf{KEM}}(m_0^{(I)})$	Parse $m_0^{(I)} = PK_E^A$. Compute $(ct_2, ss) \leftarrow \mathsf{Encaps}(PK_E^A)$. Compute $C_2 \leftarrow \mathsf{Enc}_{ss}(\mathsf{payload}_R)$. Return $m_2 = (ct_2, C_2)$ and $st_1^{(R)} = (ss)$.
$\mathsf{DHInit}^{\mathsf{KEM}}(st_0^{(I)}, m_2)$	Parse inputs. Recover $ss \leftarrow \mathsf{Decaps}(SK_E^A, ct_2)$. Decrypt C_2 to process payload. Compute $C_3 \leftarrow \mathsf{Enc}_{ss}(\mathsf{payload}_I)$. Return $m_1 = (C_3)$ and $st_1^{(I)} = (ss)$.
$\mathsf{Compi}^{\mathsf{KEM}}(st_1^{(I)}$ $st_0^{(I)}, m_0^{(I)}, m_1, m_2)$	Parse state for keys. Output $K_2 = (ss)$.
$\mathsf{Compr}^{\mathsf{KEM}}(st_1^{(R)}, m_0^{(I)}, m_1, m_2)$	Parse m_1 for C_3. Decrypt C_3 to process payload. Output $K_2 = (ss)$.

D Proofs

D.1 Proof of Lemma 1

Proof. The proof proceeds by replacing the honest protocol transcript with a simulated transcript, and then constructing a reduction $\mathcal{B}$ that solves the Gap-CDH problem using the adversary $\mathcal{A}$.

By Definition 7, the simulated transcript replaces the structured message A with a uniformly random element in $\mathbb{G}$. Under the DDH assumption, this simulated transcript is computationally indistinguishable from the real transcript. Therefore, transitioning to this simulated transcript alters $\mathcal{A}$'s success probability by at most $\mathsf{Adv}_{\mathsf{DDH}}(\mathcal{A}')$.

$\mathcal{B}$ receives a Gap-CDH challenge tuple $(U, V) = (g^u, g^v)$ and has access to a decision oracle $\mathsf{DDH}(g, \cdot, \cdot, \cdot)$. $\mathcal{B}$'s goal is to compute g^{uv}.

Transcript Generation. $\mathcal{B}$ starts $\mathcal{A}$ and receives the target password pw. If $pw = 0$, $\mathcal{B}$ aborts. $\mathcal{B}$ programs the J-PAKE session to embed the challenges U and V as follows:

- Set $X_2 := U = g^u$. This implicitly sets the Initiator's secret $x_2 = u$, which $\mathcal{B}$ does not know.
- Sample $x_1 \xleftarrow{\$} \mathbb{Z}_q$ and calculate $X_1 = g^{x_1}$.
- Set $X_3 := V \cdot X_1^{-1}$. This implicitly sets $X_1 X_3 = V$.
- Sample $x_4 \xleftarrow{\$} \mathbb{Z}_q$ and set $X_4 := g^{x_4}$. $\mathcal{B}$ explicitly knows x_4.

$\mathcal{B}$ generates the remaining messages using the known parameters pw and x_4:

$$B := (X_1 X_2 X_3)^{x_4 \cdot pw} = (V \cdot U)^{x_4 \cdot pw}, \quad A \xleftarrow{\$} \mathbb{G}.$$

$\mathcal{B}$ simulates the ZK proofs $(\pi_1, \pi_2, \pi_3, \pi_4, \pi_A, \pi_B)$ using the SE-NIZK simulator and sends the transcript to $\mathcal{A}$. Note that substituting the real proofs with simulated ones alters the adversary's advantage by at most $\mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIZK}}(\mathcal{A}'')$.

Oracle Simulation. $\mathcal{A}$ makes adaptive queries to verify keys. $\mathcal{B}$ must answer them without knowing x_2.

Oracle $\mathcal{O}_1$ (Initiator Key): On input (pw', K'), the exact equation for the correct initiator key is $K = (X_1 X_3)^{x_2 x_4 pw'}$. Substituting the embedded values yields $K = (V)^{x_2 \cdot x_4 \cdot pw'} = (V^{x_4 \cdot pw'})^{x_2}$. $\mathcal{B}$ calculates the base $Z := V^{x_4 \cdot pw'}$. The oracle must check if $K' = Z^{x_2}$. Since $\mathcal{B}$ published $X_2 = g^{x_2}$, this is mathematically equivalent to checking if (g, X_2, Z, K') is a valid Diffie-Hellman tuple. $\mathcal{B}$ calls its decision oracle and returns the output of $\mathsf{DDH}(g, X_2, Z, K')$.

Oracle $\mathcal{O}_2$ (Responder Key): Similar to $\mathcal{O}_1$, here the correct key equation for the responder is $K = (A \cdot (X_2^{pw'})^{-x_4})^{x_4}$. $\mathcal{B}$ defines the base value as $S := A \cdot (X_2^{pw'})^{-x_4}$. Now it needs to check if $K' = S^{x_4}$. Since $\mathcal{B}$ provided the public key $X_4 = g^{x_4}$, checking $K' = S^{x_4}$ is equivalent to verifying the DH tuple (g, X_4, S, K'). $\mathcal{B}$ calls its own oracle and outputs $\mathsf{DDH}(g, X_4, S, K')$.

The Responder calculates the key as $K = (A/\text{Mask})^{x_4}$. Normally, the Mask is $X_2^{x_4 \cdot pw} = g^{x_2 \cdot x_4 \cdot pw}$. Calculating this usually requires knowing x_2. However, $\mathcal{B}$ uses the trick $g^{x_2 \cdot x_4} = (g^{x_2})^{x_4} = X_2^{x_4}$ Since $\mathcal{B}$ knows x_4 (it chose it) and X_2 (the challenge U), it can calculate the Mask directly.

Solving the Challenge. If $\mathcal{A}$ wins the One-Wayness game, it outputs the correct DH-key K^* for the target password pw: $K^* = (X_1 X_3)^{x_2 x_4 pw}$. Substituting the embedded values $X_1 X_3 = V = g^v$ and $x_2 = u$, we have $K^* = (g^v)^{u \cdot x_4 \cdot pw} = g^{uv \cdot (x_4 \cdot pw)}$. Because $\mathcal{B}$ chose x_4 and was given pw, it computes the modular inverse $\alpha := (x_4 \cdot pw)^{-1} \pmod q$. Finally, $\mathcal{B}$ computes and outputs the solution to the Gap-CDH challenge $W := (K^*)^\alpha = g^{uv}$. This extracts the correct Gap-CDH solution from the adversary's output. Therefore, the adversary's total advantage in breaking the One-Wayness of the key is bounded by:

$$\mathsf{Adv}^{\mathsf{ow}}_{\mathsf{key}}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{Gap\text{-}CDH}}(\mathcal{B}) + \mathsf{Adv}_{\mathsf{DDH}}(\mathcal{A}') + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIZK}}(\mathcal{A}'') = \mathsf{negl}(\kappa).$$

D.2 Proof of Lemma 2

Proof. Fix pp, td, a password $pw \in \mathbb{Z}_q^*$, and fixed responder messages $m_0^{(R)}$ and m_2. We analyze the indistinguishability of the initiator's messages step by step.

Pre-phase message ($m_0^{(I)}$).

- **Real:** The initiator samples $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_q$ and computes $X_1 = g^{x_1}, X_2 = g^{x_2}$ and proofs π_1, π_2.
- **Simulated:** SimPreInit samples x_1, x_2 exactly as above. It generates proofs π_1', π_2' using the Zero-Knowledge simulator.

By the Zero-Knowledge property of the NIZK system, the simulated proofs are computationally indistinguishable from real proofs. Since the values X_1, X_2 are sampled identically from the uniform distribution in both worlds, the distributions of $m_0^{(I)}$ are only bounded by $\mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIZK}}(\mathcal{D}'')$.

Third-pass message (m_1). Let $G := X_1 X_3 X_4$.

- **Real:** The initiator computes $A := G^{x_2 \cdot pw}$. Note that $X_2 = g^{x_2}$ was published in Step 1 and the adversary (Environment) knows pw and G.
- **Simulated:** SimDHInit samples a random $r' \xleftarrow{\$} \mathbb{Z}_q$ and sets $A' := G^{r'}$.

We claim that distinguishing the real A from the simulated A' is equivalent to solving the DDH problem. Consider the tuple $(g, g^a, g^b, g^{ab}) := (g, X_2, G^{pw}, A)$. In the real world, $A = (G^{pw})^{x_2} = (g^b)^{x_2}$, so this is a valid DDH tuple relative to the base g. In the simulated world, A' is a random group element independent of X_2 and pw, making the tuple random. Under DDH hardness, these two views are computationally indistinguishable: $(m_0^{(I)}, m_1) \approx_c (m_0'^{(I)}, m_1')$. The advantage in

distinguishing these views is bounded by $\mathsf{Adv}_{\mathsf{DDH}}(\mathcal{D}') + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIZK}}(\mathcal{D}'')$, where the second term accounts for the simulation of π'_A.

DH-Key K. We now show that the derived keys are identical conditioned on the internal state.

- **Real:** $K = (B \cdot X_4^{-x_2 \cdot pw})^{x_2} = (X_1 X_3)^{x_2 x_4 pw}$.
- **Simulated:** The algorithm $\mathsf{SimCompi}$ takes as input the trapdoor td, the password pw, and the state $st_1^{(I)} = x_2$. It uses td to extract the witness x_4 from the responder's proof π_4. It then computes $K' = (X_1 X_3)^{x_2 \cdot x_4 \cdot pw}$.

Since the simulator uses the identical x_2 (from Step 1) and the correct x_4 (via extraction), the calculation of K' is mathematically identical to K. Thus, conditioned on the previous steps, the key distribution is identical (advantage is 0). Combining the ZK property (Step 1), the DDH indistinguishability (Step 2), and the identical key derivation (Step 3), the joint distribution is computationally indistinguishable.

D.3 Proof of Lemma 3

Proof. The proof for this lemma is similar to the proof for Lemma 2. The only difference is that we need to fix the initiator's messages and simulate for the responder.

D.4 Proof of Lemma 4

Proof. By Lemma 2, replacing the honest transcript with a simulated transcript alters the adversary's advantage by at most $\mathsf{Adv}_{\mathsf{DDH}}(\mathcal{A}') + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIZK}}(\mathcal{A}'')$. We construct a reduction algorithm $\mathcal{B}$ that solves the $\mathsf{Gap\text{-}CDH}$ problem using any adversary $\mathcal{A}$ operating on this simulated transcript.

$\mathcal{B}$ receives a challenge tuple $(U, V) = (g^a, g^b)$ and has access to a Decision Diffie-Hellman oracle $\mathsf{DDH}(\cdot,\cdot,\cdot,\cdot)$. $\mathcal{B}$'s goal is to compute $W = g^{ab}$.

Simulated Transcript. $\mathcal{B}$ generates $(pp^*, td^*) \leftarrow \mathsf{DHSetup}(1^\kappa)$ and acts as the Initiator in the challenge session. $\mathcal{B}$ embeds the challenges U and V into the first message $m_0^{(I)*}$:

- Set $X_2^* := U = g^a$. (This implicitly sets the secret $x_2 = a$, which $\mathcal{B}$ does **not** know).
- Set $X_1^* := V = g^b$. (This implicitly sets $x_1 = b$).
- Use the SE-NIZK simulator to generate proofs π_1^*, π_2^*.

$\mathcal{B}$ sends $m_0^{(I)*} = (X_1^*, X_2^*, \pi_1^*, \pi_2^*)$ to $\mathcal{A}$. When required, $\mathcal{B}$ generates the third-pass message m_1^* using $\mathsf{SimDHInit}$ (sampling a random A^* and simulating π_A^*).

Perfect Simulation of Oracles. $\mathcal{A}$ makes verification queries. $\mathcal{B}$ answers these using the DDH oracle when x_2 is unknown.
Oracle $\mathcal{O}_1^*$ (Challenge Session): Given a password guess pw' and candidate key K', the correct relation is $K = (X_1^* X_3)^{x_2 x_4 pw'}$.

1. $\mathcal{B}$ parses $m_0^{(R)}$ to find π_4 and extracts the witness: $x'_4 \leftarrow \mathsf{Ext}(td^*, \pi_4)$.
2. $\mathcal{B}$ computes the base $Z := (X_1^* X_3)^{x'_4 \cdot pw'}$. Note X_1^*, X_3, x'_4, pw' are all known to $\mathcal{B}$.
3. The check reduces to whether $K' \stackrel{?}{=} Z^{x_2}$. Since $X_2^* = g^{x_2}$, this is equivalent to checking if (g, X_2^*, Z, K') is a DDH tuple.
4. $\mathcal{B}$ returns the output of $\mathsf{DDH}(g, X_2^*, Z, K')$.

Oracle $\mathcal{O}_1$ and $\mathcal{O}_2$ (Other Sessions) For queries regarding other sessions (non-challenge), $\mathcal{B}$ also knows the trapdoor td^*.

- If $\mathcal{B}$ is acting as the Initiator (Oracle $\mathcal{O}_1$), it extracts x_4 from the adversary's proof, computes $Z = (X_1 X_3)^{x_4 pw'}$, and checks $\mathsf{DDH}(g, X_2, Z, K')$. (Even if $\mathcal{B}$ knows x_2 in other sessions, using the oracle is valid).
- If $\mathcal{B}$ is acting as the Responder (Oracle $\mathcal{O}_2$), it extracts x_2 from the adversary's proof π_2. It computes the base $S = A \cdot (X_2^{pw'})^{-x_4}$. It checks if $K' = S^{x_4}$ by querying $\mathsf{DDH}(g, X_4, S, K')$.

Solving the Gap-CDH Problem. Eventually, $\mathcal{A}$ outputs a success tuple $(pw, pw', m_0^{(R)}, m_2, K, K')$ where $pw \neq pw'$. Since $\mathcal{A}$ won, both keys must be valid with respect to the challenge transcript $m_0^{(I)*}$, implying:

$$K = (X_1^* X_3)^{x_2 x_4 pw} \quad \text{and} \quad K' = (X_1^* X_3)^{x_2 x_4 pw'}$$

$\mathcal{B}$ performs the following extraction:

1. $\mathcal{B}$ extracts x_3 and x_4 from π_3 and π_4.
2. $\mathcal{B}$ computes the ratio $\frac{K}{K'} = (X_1^* X_3)^{x_2 x_4 (pw - pw')}$.
3. Since $pw \neq pw'$, $\Delta = (pw - pw')$ is invertible mod q. $\mathcal{B}$ computes the inverse $\alpha = (x_4 \cdot \Delta)^{-1} \pmod q$[4].
4. $\mathcal{B}$ computes $Y := \left(\frac{K}{K'}\right)^{\alpha} = (X_1^* X_3)^{x_2}$.
5. Substituting $X_1^* = g^b$, $X_2^* = g^a$ ($x_2 = a$), and $X_3 = g^{x_3}$, we have $Y = (g^b \cdot g^{x_3})^a = g^{ba} \cdot g^{a x_3}$.
6. $\mathcal{B}$ isolates g^{ab} by computing the noise term $(X_2^*)^{x_3} = g^{a x_3}$.
7. $\mathcal{B}$ outputs the $\mathsf{Gap\text{-}CDH}$ solution: $W^* := Y \cdot ((X_2^*)^{x_3})^{-1} = g^{ab}$.

Thus, the adversary's advantage is bounded by:

$$\mathsf{Adv}^{\mathsf{Init}}_{\mathsf{Unipw}}(\mathcal{A}) \leq \mathsf{Adv}_{\mathsf{Gap\text{-}CDH}}(\mathcal{B}) + \mathsf{Adv}_{\mathsf{DDH}}(\mathcal{A}') + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIZK}}(\mathcal{A}'') = \mathsf{negl}(\kappa)$$

D.5 Proof of Lemma 5

Proof. This proof is a symmetry of Lemma 4's proof.

[4] Note that $x_4 \neq 0 \pmod q$ because the protocol aborts if $X_4 = 1$, ensuring the extracted x_4 is invertible.

D.6 Proof of Theorem 1

Proof. The proof follows the modular framework established in the proof of Theorem 1 in Lyu and Liu [34], demonstrating that the real-world execution is indistinguishable from the ideal simulation via a sequence of games (G_0 to G_8). The reduction relies on the One-Wayness of the DH-key (for passive security), the Unique Password property (for active security), and the Computational Simulation of transcripts. Specifically, the transitions from Game 2 to 3 and Game 3 to 4 replace honest initiator and responder messages with simulated ones. Unlike the original framework's perfect simulation, these steps differ by the computational advantage of distinguishing the simulation, bounded by $\ell \cdot \mathsf{Adv}^{\mathsf{Init}}_{\mathsf{Sim}}$ and $\ell \cdot \mathsf{Adv}^{\mathsf{Resp}}_{\mathsf{Sim}}$, respectively. Therefore, we highlight the specific modifications required to accommodate the computational simulation property of J-PAKE and derive the resulting security bound.

- **Games $G_0 \to G_2$ (Passive Security):** These games replace the real session keys with random keys for passively observed sessions. As in [34], the indistinguishability relies on the *One-Wayness* of the DH-key (Lemma 1). The distinguishing advantage is bounded by $\ell^2 \cdot \mathsf{Adv}^{\mathsf{ow}}_{\mathsf{key}}(\mathcal{B})$, where ℓ is the maximum number of sessions.
- **Games $G_2 \to G_3$ (Simulating Initiator Messages):** In this step, the simulator replaces the real initiator messages m_1 generated by DHInit with simulated messages generated by $\mathsf{SimDHInit}$. In the original framework, *perfect simulation* implied these games were identical ($Pr[G_3] = Pr[G_2]$). For J-PAKE, we apply Lemma 2. By the hardness of the DDH problem, the real and simulated transcripts are computationally indistinguishable. Across ℓ potential sessions, the difference is bounded by $|\Pr[G_3] - \Pr[G_2]| \leq \ell \cdot \mathsf{Adv}^{\mathsf{Init}}_{\mathsf{Sim}}$.
- **Games $G_3 \to G_4$ (Simulating Responder Messages):** Similarly, the simulator replaces real responder messages m_2 with outputs from $\mathsf{SimDHResp}$. By Lemma 3 and the DDH assumption, the difference is bounded by $|\Pr[G_4] - \Pr[G_3]| \leq \ell \cdot \mathsf{Adv}^{\mathsf{Resp}}_{\mathsf{Sim}}$.
- **Games $G_4 \to G_6$ (Active Security / Unique Password):** These games introduce abort conditions if the adversary successfully produces valid DH-keys for two different passwords in the same session. As in [34], this is bounded by the *Unique Password* property (Lemma 4 and 5), which is $\ell \cdot (\mathsf{Adv}^{\mathsf{Init}}_{\mathsf{Unipw}} + \mathsf{Adv}^{\mathsf{Resp}}_{\mathsf{Unipw}})$.
- **Games $G_6 \to G_8$ (Ideal Functionality):** These steps are conceptual changes to interface with $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$ and involve no change in distribution.

Combining the advantages from the sequence of games, the total advantage of any environment $\mathcal{Z}$ in distinguishing the real protocol Π from the ideal functionality $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$ is bounded by:

$$\mathsf{Adv}^{\mathsf{UC}}_{\mathcal{Z}} \leq \underbrace{\ell^2 \cdot \mathsf{Adv}^{\mathsf{ow}}_{\mathsf{key}}}_{\text{Passive Security}} + \underbrace{\ell \cdot \left(\mathsf{Adv}^{\mathsf{Init}}_{\mathsf{Unipw}} + \mathsf{Adv}^{\mathsf{Resp}}_{\mathsf{Unipw}}\right)}_{\text{Active Extraction}} + \underbrace{\ell(\mathsf{Adv}^{\mathsf{Init}}_{\mathsf{Sim}} + \mathsf{Adv}^{\mathsf{Resp}}_{\mathsf{Sim}})}_{\text{Transcript Simulatability}}.$$

Since all terms on the right-hand side are negligible for a PPT adversary under the Gap-CDH and DDH assumptions, the total advantage is negligible. Thus, Π UC-emulates $\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$.

D.7 Proof of Lemma 6

Proof. We define the simulation algorithms for the KEM component by setting them to be exactly the honest protocol algorithms: $\mathsf{SimPreInit}^{\mathsf{KEM}} := \mathsf{PreInit}^{\mathsf{KEM}}, \mathsf{SimDHInit}^{\mathsf{KEM}} := \mathsf{DHInit}^{\mathsf{KEM}}, \mathsf{SimCompi}^{\mathsf{KEM}} := \mathsf{Compi}^{\mathsf{KEM}}$. This is a valid simulation strategy because the core KEM component is *password-independent*. The honest algorithms do not utilize the password pw in any internal computation. Thus, the simulator trivially satisfies the requirement of generating the transcript view without knowledge of pw (treating the external payload as a fixed input string). The detailed analysis of the distributions is as follows:

1. **Pre-Phase ($m_0^{(I)}$):** The real initiator runs KeyGen to obtain (PK_E^A, SK_E^A). The simulator executes the exact same KeyGen algorithm using freshly sampled randomness. Therefore, the distribution of the public keys is mathematically identical.
2. **Third-Pass (m_1):** The real initiator decapsulates ss from m_2 and uses it as a symmetric key to encrypt the payload pl (representing the J-PAKE messages), yielding ciphertext C_3. The simulator executes the exact same symmetric encryption routine. Conditioned on the same fixed input payload pl, the outputs are identically distributed over the randomness of the encapsulation and encryption schemes.
3. **Key (K_2):** In both the real and simulated experiments, the DH-key is derived as $K_2 = (ss)$, which is the direct deterministic output of the honest decapsulation routine given the transcript.

Since the simulator executes identical probabilistic polynomial-time algorithms as the honest party using the same distributions of randomness and inputs, the joint distribution of the transcript and key is perfectly identical:

$$(m_0^{(I)}, m_1, m_2, K_2) \equiv (m_0'^{(I)}, m_1', m_2, K_2')$$

Remark 3 (Hybrid Argument for the Full Protocol). In the context of the full Hybrid PAKE proof, the payload pl will differ between worlds (Real J-PAKE messages vs. Simulated J-PAKE messages). However, the indistinguishability of the composite protocol holds via a standard hybrid argument: first, the underlying J-PAKE payload is replaced with its simulation (computationally indistinguishable by Lemma 2 under the DDH assumption), and then the KEM wrapper is applied (perfectly identical by this lemma).

D.8 Proof of Lemma 7

Proof. Analogous to Lemma 6, we define the responder's simulation algorithms $(\mathsf{SimDHResp}^{\mathsf{KEM}}, \mathsf{SimCompr}^{\mathsf{KEM}})$ to be identical to the honest responder algorithms. Since the protocol is password-independent, the simulator produces a transcript and key $K_2 = (ss)$ that are identically distributed to the real execution.

D.9 Proof of Lemma 8

Proof. We prove this lemma via a sequence of games, constructing two reduction algorithms $(\mathcal{B}_1, \mathcal{B}_2)$ to bound the indistinguishability based on the IND-CCA2 security of the KEM and the IND-CPA security of the symmetric encryption. Let S_i be the event that $\mathcal{D}$ outputs 1 in Game_i.

Game 0 (Real Execution). The protocol is executed honestly.

- The KEM keypairs (PK_E, SK_E) are generated honestly.
- The shared secret ss is derived via honest encapsulation.
- The ciphertexts C_2, C_3 encrypt the real J-PAKE payloads M_2, M_3 (containing X_3, X_4, A, B, and ZK proofs) using the key derived from ss.
- The final output key is $K_2 = ss$.

By definition, the probability of S_0 is $\Pr[S_0] = \Pr[\mathcal{D}(\mathsf{View}^{\mathsf{Real}}) = 1]$.

Game 1 (Key Replacement / KEM Security). In this game, we replace the KEM-derived shared secret with a truly uniform random string. Instead of using the real ss output by Encaps, we sample $ss^* \leftarrow \$\mathcal{K}$.

- The ciphertexts C_2, C_3 are encryptions of the **real payloads** M_2, M_3, but generated using the **random key** ss^*.
- The final output key is $K_2 = ss^*$.

The KEM ciphertext ct_2 is still generated honestly, but it effectively encapsulates "garbage" relative to the random key used for the payload encryption.
Claim 1: $|\Pr[S_0] - \Pr[S_1]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA2}}_{\mathsf{KEM}}(\mathcal{B}_1)$.
Proof of Claim 1: We construct a reduction $\mathcal{B}_1$ against the IND-CCA2 security of the KEM. $\mathcal{B}_1$ receives a challenge (ct^*, K^*) from its challenger. It embeds ct^* into the protocol transcript as ct_2 and sets the symmetric encryption key $ss^* := K^*$. If K^* is the real encapsulated key, the distribution perfectly matches Game 0. If K^* is a random key, the distribution perfectly matches Game 1. Thus, $\mathcal{D}$'s advantage bounds $\mathcal{B}_1$'s advantage.

Game 2 (Payload Replacement / Symmetric Security). In this game, we modify the generation of the symmetric ciphertexts. Since the key ss^* is now a truly random string (from Game 1), we invoke the indistinguishability of the symmetric cipher. Instead of encrypting the real payloads M_2, M_3, the game encrypts strings of zeros of equal length:

$$C_2' \leftarrow \mathsf{Enc}_{ss^*}(0^{|M_2|}) \quad \text{and} \quad C_3' \leftarrow \mathsf{Enc}_{ss^*}(0^{|M_3|})$$

The final output key remains random ($K_2 = ss^*$). This view is mathematically identical to $\mathsf{View}^{\mathsf{Sim}}$.
Claim 2: $|\Pr[S_1] - \Pr[S_2]| \leq 2 \cdot \mathsf{Adv}_{\mathsf{SE}}^{\mathsf{IND\text{-}CPA}}(\mathcal{B}_2)$.
Proof of Claim 2: We construct a reduction $\mathcal{B}_2$ against the IND-CPA security of the symmetric encryption scheme. $\mathcal{B}_2$ submits the real payloads and dummy payloads $(M, 0^{|M|})$ to its challenger. Because the key used by the challenger is uniformly random (matching Game 1), the reduction is valid. The factor of 2 accounts for replacing the two ciphertexts (C_2, C_3) via a standard hybrid argument across multiple messages. If the challenger encrypts M, the view is Game 1; if it encrypts $0^{|M|}$, the view is Game 2.

Thus, the total advantage is bounded by:

$$|\Pr[S_0] - \Pr[S_2]| \leq \mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA2}}(\mathcal{B}_1) + 2 \cdot \mathsf{Adv}_{\mathsf{SE}}^{\mathsf{IND\text{-}CPA}}(\mathcal{B}_2)$$

Since both advantage terms are negligible for PPT adversaries, the real view is computationally indistinguishable from the simulated view.

Remark 4 (Public Key Indistinguishability). The simulator for Game 2 generates the ephemeral public key PK_E^A using the honest KeyGen algorithm. Because the honest algorithm is used in both the real world and the simulation, the distribution of public keys is identical. Thus, we do not need to invoke the MLWE assumption to bound the distinguishability of the public keys themselves, only for the encapsulation (ct_2, ss).

D.10 Proof of Theorem 2

Proof. **J-PAKE is Full DH-type.** By Lemmas 1 to 5, we established that the J-PAKE component admits computational simulation of messages, one-wayness of the DH key, and unique password extraction. These properties hold under the $\mathsf{Gap\text{-}CDH}$ and DDH assumptions.

KEM is Simulatable DH-type. By Lemmas 6 and 7, we established that the KEM component admits perfect simulation of its transcripts and the derived key. This simulation property is unconditional: since the honest KEM algorithms do not take the password as input, the simulator can perfectly reproduce the distribution of a real execution (running the honest algorithms) without knowledge of the password or any private keys.

Direct Proof of Hybrid Security. While the key derivation $K_{\mathsf{session}} = \mathsf{HKDF}(K_1 \| K_2 \| \tau)$ is parallel, the transcript structure involves nesting J-PAKE messages within KEM ciphertexts. To address this, we explicitly demonstrate that the combined protocol Π_{HPAKE} satisfies the five properties required by the definition of a Full DH-type PAKE:

Computational Simulatability. We construct a hybrid simulator $\mathsf{Sim}_{\mathsf{H}}$ that utilizes the underlying J-PAKE simulator $\mathsf{Sim}_{\mathsf{J}}$ and the honest KEM algorithms. $\mathsf{Sim}_{\mathsf{H}}$ runs $\mathsf{Sim}_{\mathsf{J}}$ to generate all simulated J-PAKE messages $\tau_{\mathsf{J}}^* =$

$(m_0^{(I)*}, m_1^*, m_0^{(R)*}, m_2^*)$. It then runs the honest KEM encapsulation algorithms to produce the ciphertexts $C = \mathsf{Enc}(\tau_\mathsf{J}^*)$. Since the honest KEM encryption is a randomized functionality and the simulated J-PAKE messages are computationally indistinguishable from real messages (by DDH), the hybrid transcript is computationally indistinguishable from a real execution.

$$\mathsf{Adv}^{\mathsf{Sim}}_{\mathsf{Hybrid}} \leq \mathsf{Adv}^{\mathsf{Sim}}_{\mathsf{J-PAKE}} + \mathsf{Adv}^{\mathsf{ZK}}_{\mathsf{NIZK}} + \mathsf{Adv}^{\mathsf{IND-CCA2}}_{\mathsf{KEM}}$$

One-Wayness. We prove the session key $K_{\mathsf{session}} = \mathsf{HKDF}(K_1 \| K_2 \| \tau)$ is indistinguishable from random via a sequence of games:

- **Game 0 (Real).** Real hybrid execution.
- **Game 1 (KEM Idealization).** We replace KEM ciphertexts with encryptions of zeros (justified by IND-CCA2). This statistically removes the J-PAKE messages from the adversary's view in the transcript.
- **Game 2 (J-PAKE Simulation).** We switch J-PAKE generation to its simulator. This is indistinguishable due to the simulatability of J-PAKE.
- **Game 3 (Randomness).** The key is $K_{\mathsf{session}} = \mathsf{HKDF}(K_1 \| K_2 \| \tau)$; since K_1 possesses high min-entropy (guaranteed by the One-Wayness of J-PAKE) and is independent of the KEM component, the PRF security of HKDF ensures that the output K_{session} is indistinguishable from random, even if K_2 is known to the adversary.

Unique Password (Initiator and Responder). We must show that for a fixed simulated transcript, it is computationally infeasible for an adversary $\mathcal{A}$ to produce two distinct passwords $pw \neq pw'$ that both yield valid session keys. Let $\mathsf{Adv}^{\mathsf{Hybrid}}_{\mathsf{Unipw}}(\mathcal{A})$ be the advantage of $\mathcal{A}$ against the hybrid protocol. We construct a reduction $\mathcal{B}$ against the Unique Password property of the underlying J-PAKE component. $\mathcal{B}$ receives a J-PAKE challenge transcript τ_J^* from its challenger. $\mathcal{B}$ wraps this transcript in honest KEM ciphertexts to create a hybrid transcript τ_H^* and forwards it to $\mathcal{A}$. The hybrid key is derived as $K_{\mathsf{session}} = \mathsf{HKDF}(K_1(pw) \| K_2 \| \tau)$. Note that K_2 is fixed by the transcript and is *independent* of the password. If $\mathcal{A}$ outputs $pw \neq pw'$ such that both result in valid hybrid keys, then $\mathcal{A}$ must have found a collision in the HKDF inputs. Since K_2 is constant, this implies a collision in the J-PAKE keys: $K_1(pw)$ and $K_1(pw')$ must both be valid with respect to the J-PAKE transcript. Thus, any successful attack on the Hybrid Unique Password property translates directly to an attack on the J-PAKE Unique Password property. Hence :

$$\mathsf{Adv}^{\mathsf{Hybrid}}_{\mathsf{Unipw}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{J-PAKE}}_{\mathsf{Unipw}}(\mathcal{B}) + \mathsf{negl}(\kappa).$$

Since $\Pi_{\mathsf{J-PAKE}}$ satisfies the Unique Password property (Definitions 9 and 10), the hybrid protocol inherits this property. Thus, Π_{HPAKE} satisfies Computational Simulatability (relying on the KEM and J-PAKE), One-Wayness (relying on J-PAKE and HKDF), and Unique Password properties (by J-PAKE). It is therefore a Full DH-type PAKE. By Theorem 1 of Lyu and Liu [34], it UC-securely realizes

$\mathcal{F}_{\mathsf{bPAKE}^{\mathsf{LE}}}$. Crucially, we note that while the transcript simulation utilizes the KEM's security, the One-Wayness of the session key is maintained solely by the classical Gap-CDH and DDH assumptions of the J-PAKE component, ensuring key confidentiality even if the KEM is eventually compromised.