Dance with Noise: Safely Trade Minor Decryption Failures for More Compact KEMs with Applications to IKEv2

Zidi Zhuang[1,2], Yuyang Xiao[1,2], Long Chen[1], Qiang Tang[3], and Zhenfeng Zhang[1]

[1]Institute of Software, Chinese Academy of Sciences, China
[2]University of Chinese Academy of Sciences, China
[3]The University of Sydney, Australia

September 24, 2026

Full version of the paper accepted to ACM CCS 2026.

Abstract

The standardization of post-quantum cryptography, notably ML-KEM, introduces a severe network bottleneck: large public keys and ciphertexts often exceed the Maximum Transmission Unit (MTU) limits of UDP-based protocols. This size explosion inevitably leads to unreliable IP-layer fragmentation or forces complex protocol workarounds.

To overcome this, we propose a novel design philosophy for post-quantum Authenticated Key Exchange (AKE). We challenge the rigid cryptographic paradigm that mandates negligible decryption failure rates (DFR). By safely trading minor, manageable decryption failures for aggressive size compression, we drastically reduce the bandwidth footprint of lattice KEMs in ephemeral exchanges. We demonstrate this approach via a compact AKE for the Internet Key Exchange protocol (IKEv2). Integrating standard ML-KEM into IKEv2 currently necessitates the high-latency `IKE_INTERMEDIATE` exchange (RFC 9242) to distribute large payloads. By employing our compressed KEM design (800-byte public key, 928-byte ciphertext), the entire key exchange fits within the initial `IKE_SA_INIT` packet, completely eliminating the need for RFC 9242.

To ensure that protocol aborts caused by decryption failures do not introduce security vulnerabilities, we formalize IND-CPAF (Indistinguishability under Chosen Plaintext Attack with Failures) to rigorously bound failure-dependent leakage. More importantly, we prove our IKEv2 instantiation still secure in the classical Canetti-Krawczyk model. Finally, our implementation in the `strongSwan` IPsec library confirms a fragmentation-free handshake: over 10,000 real-world connections, it achieves a 48.8 ms average setup time—matching classical ECDH—and keeps initial packets strictly under the 1280-byte IPv6 MTU, with merely a 0.002% retry rate.

1 Introduction

The migration to Post-Quantum Cryptography (PQC) is no longer a theoretical exercise but an imminent infrastructure overhaul. While the cryptographic community has coalesced around lattice-based Key Encapsulation Mechanisms (KEMs)—culminating in the standardization of ML-KEM [26], also known as Kyber [2], by NIST [3]—the integration of these primitives into real-world protocols faces significant friction. The primary source of this friction is bandwidth: post-quantum public keys and ciphertexts are orders of magnitude larger than their classical Diffie-Hellman counterparts.

This size explosion is particularly problematic for the **Internet Key Exchange protocol version 2 (IKEv2)** [22], the standardized protocol for establishing IPsec VPNs. Unlike TLS, which runs over TCP, IKEv2 operates over UDP. This seemingly minor transport difference imposes a strict physical constraint: the *Maximum Transmission Unit (MTU)*.

Packet Size and Fragmentation Challenges. Post-quantum KEMs significantly increase the size of key exchange payloads in IKEv2. When these payloads are carried in UDP packets, the resulting messages may *exceed* the path MTU, triggering packet fragmentation. The resulting behavior differs between IPv4 and IPv6:

- **IPv4: Router-assisted fragmentation.** IPv4 routers can fragment oversized packets, enabling large UDP packets to traverse the network without sender-side handling. However, fragmented traffic is often unreliable in practice, as many middleboxes (e.g., firewalls, NAT devices, load balancers) drop non-initial fragments [29]. Consequently, fragmented IKE messages may fail to reach the peer.

- **IPv6: Source fragmentation or PMTU feedback.** IPv6 routers do not perform fragmentation. Instead, packets exceeding the path MTU trigger an ICMPv6 "Packet Too Big" message, allowing the sender to adjust its packet size using Path MTU Discovery (PMTUD). Alternatively, the sending host may perform source fragmentation by inserting an IPv6 Fragment header. In practice, however, PMTUD can fail when ICMP messages are filtered, and fragmented IPv6 traffic is often treated similarly to fragmented IPv4 traffic by middleboxes.

Regardless of whether fragmentation occurs at the network layer or is performed by the sender, fragmented packets are generally less reliable than unfragmented ones. They increase the likelihood of packet loss because the loss of any fragment causes the entire message to be discarded, and they may also expose the receiver to resource exhaustion attacks on reassembly buffers.

To mitigate these issues, IKEv2 provides mechanisms to avoid relying on IP-layer fragmentation. However, its fragmentation support differs across handshake stages. IKE fragmentation, defined in RFC 7383 [29], applies only to encrypted messages *after* the IKE_SA_INIT exchange. The IKE_SA_INIT messages themselves, which carries key exchange materials, are unencrypted and thus cannot use IKE fragmentation. RFC 7296 [22] does not specify any mechanism for fragmenting them at the IKE layer.

To mitigate the size inflation caused by post-quantum key exchange, the IETF later introduced the IKE_INTERMEDIATE exchange (RFC 9242 [30]). This mechanism inserts additional exchange rounds between IKE_SA_INIT and IKE_AUTH to carry oversized key exchange payloads, instead of placing all cryptographic payloads into a single IKE_SA_INIT message. Still, this approach implicitly assumes that the IKE_SA_INIT exchange itself can be transmitted without fragmentation. Although exceeding the path MTU does not affect the cryptographic security of the key exchange, it requires either IP-layer fragmentation or additional pre-IKE_AUTH exchanges such as IKE_INTERMEDIATE, incurring extra communication and performance overhead and highlighting a fundamental efficiency–robustness trade-off in post-quantum IKE. Moreover, IKE_INTERMEDIATE authenticates later exchanges using a stage key derived solely from IKE_SA_INIT. If that exchange contains only classical ECDH, the stage key is not post-quantum, so the extra rounds do not provide quantum-era protection against unauthenticated extra payloads.

To eliminate the need for fragmentation or multi-round workarounds, the KEM public keys and ciphertexts must be drastically reduced in size.

Yet, shrinking these parameters directly conflicts with the mathematical constraints of lattice-based schemes, where smaller sizes naturally induce higher decryption failure rates. This exposes the core reason behind the bandwidth bottleneck.

The Cost of Perfection. Standard IND-CCA secure KEMs like ML-KEM pay a massive bandwidth penalty to guarantee that decryption *never* fails. This penalty stems from the fundamental mechanics of Learning with Errors (LWE). In LWE-based schemes, security relies on injecting a small random noise e into the algebraic structure (e.g., masking the secret $\mathbf{s}$ as $\langle \mathbf{a}, \mathbf{s} \rangle + e$). This creates an inherent, strict three-way trade-off among correctness, security, and efficiency:

1. *Correctness*: Decapsulation fails if the accumulated noise exceeds a certain threshold. Ensuring a negligible Decryption Failure Rate (DFR) requires a very small noise-to-modulus ratio ($\|e\|/q$).
2. *Security*: Conversely, cryptographic hardness grows with a larger noise ratio $\|e\|/q$ and a higher lattice dimension n.
3. *Efficiency*: Bandwidth and computational costs scale directly with the dimension n and modulus q.

To simultaneously maintain high security and a negligibly small DFR, KEM designers are usually forced to resolve the conflict between (1) and (2) by compromising (3). Specifically, they have to choose a combination of relatively small $\|e\|/q$ and large n or adopting conservative compression parameters, in order to achieve security goals. This, however, is at a cost of larger public key and ciphertext sizes. When these standard KEMs are deployed in IKEv2, *this size inflation collides directly with the strict IPv6 MTU limits.*

Our size calculations (Table 1) and real-world measurements (Table 2) confirm this: even the recommended (rather than the highest) level parameters, namely ML-KEM-768 (1184-byte public key) exceeds the IKE_SA_INIT payload budget, forcing IP fragmentation under the minimum 1280-byte IPv6 MTU. This leads to a grim but inevitable status: as long as standard methods insist on overwhelming correctness, it is almost hopeless to avoid network fragmentation. The requirement for "zero" decryption failures often pushes payload sizes closer to the limits of a single packet, making fragmentation much more likely in practice. Although there may exist other workarounds, such as IKE_INTERMEDIATE (RFC 9242[30]) it inevitably introduces additional complexity and latency. While this "perfect correctness" is mandatory for static encryption, where the secret key is used for an

extended period across multiple instances, we argue that it is unnecessarily restrictive for ephemeral key exchange protocols like IKEv2, which were designed to handle occasional network packet loss and handshake failures via simple retransmissions. If we can back the relaxation of DFR constraint by rigorous provable security (rather than mere heuristics), the "wasted" noise budget can be reclaimed, e.g. by choosing aggressive compress parameters for public key and ciphertext.

Yet, simply allowing failures without rigorous treatment is dangerous. On the one hand, proceeding with a failed decapsulation yields flawed session keys that may then be used to protect subsequent messages. On the other hand, if the protocol aborts upon failure, this observable reaction acts as a *validity oracle*. Since lattice failures depend on the secret key's geometry, such leakage can compromise the underlying ephemeral secrets. This motivates our central research question:

Can we utilize minor-failure KEMs in IKEv2 to fit its handshake messages into a single UDP packet, without compromising IKEv2's original security?

1.1 Contributions

In this work, we bridge the gap between theoretical lattice cryptography potentials and their real-world availability, i.e. **allowing KEM failures without triggering cascading failures among other cryptographic primitives that interact within the protocol**. We also show that, by carefully managing such imperfections while satisfying the strengthened security requirements, we can **securely** achieve efficiency gains unattainable under standard KEM design paradigms, while genuinely preserving all properties of the original protocol, with no changes to the message flow and only moderate modifications to the state machine. Our specific contributions include:

- **Theoretical Design for Error-Prone AKE:** We identify that standard IND-CPA security is insufficient for KEM-based Authenticated Key Exchange (AKE) when decryption failures are noticeable. To securely integrate error-prone KEMs, we establish two essential cryptographic requirements. First, we formalize **Rigidity**, requiring the decapsulator to deterministically detect internal decryption errors and explicitly output a failure signal, thereby preventing flawed session keys from silently propagating. Second, because this explicit rejection inherently introduces observable side-channel leakage (e.g., protocol

aborts), we introduce **IND-CPAF** (Indistinguishability under Chosen Plaintext Attack with Failure). This novel security notion ensures that encapsulated secrets remain pseudorandom even when the adversary has access to a failure validity oracle. More importantly, we formally prove that the encrypted IKEv2 (SIGMA) protocol preserves full security in the Canetti-Krawczyk (CK) model when instantiated with a KEM that is both rigid and IND-CPAF secure.

- **Compact Black-Box KEM Construction:** We provide a generic, black-box transformation to construct a rigid and IND-CPAF secure KEM from any imperfectly correct PKE with IND-CPA security. Although our construction applies to any imperfectly-correct lattice-based PKE/KEM satisfying the described security requirement, we chose ML-KEM as an example, which is the current NIST standard and provides an immediately relevant deployment target. Applying this methodology to ML-KEM results in aggressively compressed sizes which fit real-world network constraints. Specifically, our compressed scheme achieves an 800-byte public key and 928-byte ciphertext for NIST Level-3 security, and 1184-byte public key / 1152-byte ciphertext for Level-5 security. In stark contrast, standard ML-KEM requires 1184-byte / 1088-byte (Level-3) and 1568-byte / 1568-byte (Level-5) key and ciphertext sizes (Table 1). Under the *IPv6 mandatory minimum MTU of 1280 bytes*, standard ML-KEM-768 (Level-3) inevitably exceeds the packet limit and triggers IP fragmentation, while our scheme fits seamlessly without fragmentation. Under the practical 1500-byte MTU measured in cloud environments, standard ML-KEM-1024 (Level-5) still suffers from unavoidable fragmentation, whereas our optimized implementation fits entirely within a single unfragmented packet. By fitting both Level-3 (under 1280-byte MTU) and Level-5 (under 1500-byte MTU) into unfragmented UDP packets, our compact construction *completely eliminates the need for the high-latency `IKE_INTERMEDIATE` exchange (RFC 9242)* required by standard ML-KEM across all mainstream security levels.

- **Real-World System Evaluation:** We implement our scheme in the open-source IPsec library *strongSwan* [1] and evaluate it across four intercontinental cloud links. Over 10,000 real-world connections, our post-quantum variant achieves a 48.8 ms average setup time, matching classical ECDH performance, while keeping initial packets under the

[1] `https://github.com/strongswan/strongswan`

Table 1: Captured IPv6 Packet Net Size (Bytes) in IKEv2 for mainstream lattice-based KEMs (IPv6+UDP headers 48B excluded). Two critical network thresholds: (1) IPv6 mandatory minimum MTU=1280B → net payload=1232B (global universal standard); (2) Practical MTU=1500B (Alibaba Cloud) → net payload=1452B (mainstream Ethernet/cloud deployment). **Negative spare** = IP fragmentation required; Positive spare = unfragmented transmission. All KEM sizes are taken from NIST FIPS 203 and official algorithm specifications. *NTRU security levels are taken from "Non-local model" in their document for fair comparison.

Security Level	KEM Scheme	Initiator → Responder				Responder → Initiator			
		$\|pk\|$	SA_INIT	spare_1280	spare_1500	$\|ct\|$	SA_INIT	spare_1280	spare_1500
Level 1	ecp256 [31]	64	272	960	1180	64	305	927	1147
	ML-KEM-512 [26]	800	1008	224	444	768	1009	223	443
	LightSABER [9]	672	880	352	572	736	977	255	475
	NTRU-HPS-2048-677 [7]*	930	1138	94	314	930	1171	61	281
	NTRU-HRSS-701 [7]*	1138	1346	**-114**	106	1138	1379	**-147**	73
	FrodoKEM-640 [1]	9616	9824	**-8592**	**-8372**	9720	9961	**-8729**	**-8509**
	Ours	544	752	480	700	672	913	319	539
Level 3	ecp384 [31]	96	304	928	1148	96	337	895	1115
	ML-KEM-768 [26]	1184	1392	**-160**	60	1088	1329	**-97**	123
	SABER [9]	992	1200	32	252	1088	1329	**-97**	123
	NTRU-HPS-4096-821 [7]*	1230	1438	**-206**	14	1230	1471	**-239**	**-19**
	Ours	800	1008	224	444	928	1169	63	283
Level 5	ecp521 [31]	132	340	892	1112	132	373	859	1079
	ML-KEM-1024 [26]	1568	1776	**-544**	**-324**	1568	1809	**-577**	**-357**
	FireSABER [9]	1312	1520	**-288**	**-68**	1472	1713	**-481**	**-261**
	Ours	1184	1392	**-160**	60	1152	1393	**-161**	59

1280-byte IPv6 MTU with a mere 0.002% decryption failure retry rate. This negligible overhead has no measurable impact on performance, and the minimal cost of occasional retransmissions is offset by the elimination of IP fragmentation penalties.

Critically, the performance gap widens under poor network conditions: on the lossy UK-SH link (link (d) in Figure 6), standard ML-KEM-1024 suffered **32 complete handshake failures** due to fragmented packet loss, while our unfragmented implementation experienced only **1 failure**. Box plot analysis confirms this trend: except for the low-latency BJ-SH link (link (a)), standard ML-KEM shows a significantly upward-shifted latency distribution, with higher median and 99th-percentile setup times compared to our scheme. This confirms that "dancing with noise" yields a fragmentation-free, efficient post-quantum VPN handshake with negligible real-world overhead and improved robustness in challenging network environments.

1.2 Technical Overview

Our primary objective is to enable a near black-box replacement of the KEM in SIGMA-based protocols (specifically IKEv2) with a post-quantum lattice-based KEM. A major deployment hurdle for standard ML-KEM (Kyber) in IKEv2 is that its public key and ciphertext sizes, when combined with protocol's necessary overhead such as headers, often exceed the IPv6 minimum Maximum Transmission Unit (MTU). This leads to undesirable IP fragmentation or requiring complex protocol workarounds like IKE_INTERMEDIATE (RFC 9242).

To overcome this bottleneck, we propose a concrete parameterization of ML-KEM that aggressively compresses the public key and ciphertext to fit entirely within a single IPv6 MTU. The theoretical cost of this compression is an increased, non-negligible Decryption Failure Rate (DFR) of approximately 2^{-12}. However, it fundamentally violates the DFR requirement mandatory in the proof that FO-transformed PKE achieves IND-CCA secure KEM, despite the fact that such DFR is practically negligible for network usability—requiring a handshake retry only once in roughly 4000 connections.

Our work bridges this gap by absorbing the complexities of decryption failures within the cryptographic algorithms themselves, requiring only minimal, localized adjustments to the protocol state machine. Achieving this secure integration follows a logical progression from identifying vulnerabilities to formalizing and constructing the solution.

Detecting Failures via Rigidity. If our compressed KEM occasionally fails, it must explicitly report this failure rather than silently outputting an erroneous session key. A common misconception in protocol design is that silent KEM errors are benign because they will naturally be caught during the protocol's subsequent authentication phase, where the derived keys are used to compute Message Authentication Codes (MACs). However, relying on this downstream MAC verification is cryptographically unsound. The standard unforgeability of a MAC is assured *only* when its key is uniformly random, while a silently failed decapsulation may yield a predictable or highly biased key. If an active adversary forces such a failure, they could potentially compute the weak MAC key, forge the authentication tag, and completely bypass the protocol's identity verification.

Hence, we require that the decapsulator must deterministically detect decryption failures *internally*. We formalize this property as **Rigidity**. To construct a rigid scheme, we embed a verification tag $\tau = \mathsf{PRF}(m, ct)$ that is transmitted in *plaintext* (as encrypting it would subject it to the same

failure probability). The decapsulator verifies this tag and explicitly outputs a failure signal (e.g., $\perp$ or `false`) if a mismatch occurs, preventing the faulty key from silently propagating to the protocol layer.

The Insufficiency of IND-CPA. While Rigidity successfully prevents the adoption of erroneous keys, its observable behavior in an AKE protocol introduces a new attack vector. In standard AKE models, the adversary can monitor whether the protocol proceeds normally or triggers a retry. This reaction acts as an observable Failure Oracle. Because lattice-based decryption failures are tied to the geometry of the secret key [14, 4], this oracle leaks distributional information. Crucially, this leakage introduces a distributional bias on the encapsulated secret m. This means that even in a *successful* key agreement, the underlying secret is no longer uniformly random. This leads to two critical consequences: first, the final session key loses its indistinguishability from a truly random string; second, because this biased secret m is also used to derive the verification tag ($\tau = \mathsf{PRF}(m, ct)$), the security of the tag itself may be compromised. Consequently, standard IND-CPA security is fundamentally insufficient to guarantee the security of the AKE under these conditions.

The IND-CPAF Notion and Construction. To accurately capture the adversary's capabilities in the AKE setting, we introduce a new security notion: **IND-CPAF** (Indistinguishability under Chosen Plaintext Attack with Failure). This definition explicitly grants the adversary access to a *Failure Oracle*, modeling their ability to learn whether a ciphertext decrypts correctly or not (via observing the protocol's reaction), while their ultimate goal remains to distinguish the encapsulated session key from random.

Constructing a KEM that achieves IND-CPAF requires mitigating leakage from the failure oracle. Our core observation is that in AKE protocols, the KEM keys are *ephemeral*. This limits the adversary's ability to probe for decryption failures to essentially a single interaction per key pair. Because the leakage is heavily bounded (at most one bit of success/failure info per ephemeral key pair), the encapsulated secret m retains sufficient min-entropy. To eliminate geometric bias and extract this remaining secret info, we employ a Random Oracle (RO) as a randomness extractor, deriving the final session key as $K = H(m)$.

Near Black-Box Integration via Bounded Retries. By combining these techniques, we construct a KEM that is both **Rigid** and **IND-CPAF** secure. We instantiate this with our compressed ML-KEM variant, achieving MTU-compliant packet sizes with a manageable DFR of 2^{-12}. This dual-property KEM allows for a secure, near black-box integration into IKEv2.

Rigidity ensures that mismatched values are internally caught by the KEM, while IND-CPAF guarantees that the bounded leakage from this rejection mechanism does not compromise the pseudorandomness of the successfully established keys.

At the protocol level, this shifts the burden from complex cryptographic error handling (or IP fragmentation) to a simple engineering tweak. Traditionally, KEM decapsulation always succeeds. In our modified IKEv2, when the initiator receives a `false` signal from the rigid decapsulation, instead of triggering a catastrophic handshake abort, the state machine is modified to reset and retry the handshake. Note that such retry is not an authenticated continuation of a previous handshake. It is simply a fresh IKE_SA_INIT exchange with newly generated nonces, SPIs, and ephemeral KEM key pairs. The previous attempt is discarded entirely and therefore each retry constitutes an independent protocol execution.

Given the 2^{-12} DFR, the retry is rare and practically invisible to the user. To ensure robustness and prevent infinite loops or Denial-of-Service (DoS) conditions, this retry mechanism is governed by a new configuration variable that defines a maximum retry limit. Exceeding this limit results in a standard protocol abort. This minimal modification, synergizing with our Rigid IND-CPAF KEM, achieves post-quantum security and MTU compliance with negligible engineering overhead.

However, we stress that our assumption is specific to SIGMA-style AKEs like IKEv2 with strict ephemerality, and is not in general applicable to implementations of Noise or TLS 1.3 that reuse ephemeral keys across sessions.

1.3 Related Work

PQ-WireGuard. PQ-WireGuard [21] studies compact post-quantum instantiations of the WireGuard VPN protocol [11], the latter being widely deployed due to its simplicity and efficiency. To preserve WireGuard's single-UDP-packet handshake constraint, it replaces the ephemeral KEM with a modified SABER [8] variant, *Dagger*, reducing bandwidth at the cost of increasing the decryption failure rate from $2^{-136.14}$ to $2^{-25.25}$. To compensate, PQ-WireGuard introduces an additional key-confirmation step, extending the original two-pass handshake. However, its analysis does not explicitly model risks induced by non-negligible decryption failures (e.g., bias or adversarial exploitation). This leaves open whether compact post-quantum AKE can retain the original handshake structure while rigorously accounting for such risks.

Relaxed CCA notions. Huguenin-Dumittan and Vaudenay [20] ana-

lyze TLS 1.3 [12] under IND-qCCA, a relaxation of IND-CCA introduced in KEMTLS [28], where IND-1CCA corresponds to the case $q = 1$. They provide random-oracle transformations from OW-CPA PKE to IND-qCCA KEMs. In contrast, our setting differs in three aspects: (i) their relaxation is driven by computational considerations and does not target bandwidth reduction; (ii) IND-1CCA assumes perfect correctness and does not capture decryption failures; (iii) we explicitly model failure behavior via IND-CPAF and rigidity, extending IND-CPA to regimes with non-negligible failure.

KEMs with decryption failures. A line of work studies attacks and limitations of KEMs with non-negligible failure rates, particularly under IND-CCA security and key reuse [13, 16, 17, 14]. Standard constructions rely on the Fujisaki–Okamoto (FO) transform [15, 18], typically requiring negligible failure. Recent works revisit this assumption: Hövelmanns et al. [19] tighten QROM reductions of FO under non-negligible failure, and [24] show that PVW-based schemes [27, 25] can achieve IND-CCA security despite large failure rates. These results indicate that decryption failure and IND-CCA security are not inherently incompatible, motivating a re-examination of correctness requirements in protocol design.

2 IKEv2 and Post-Quantum Migration

The Internet Key Exchange protocol version 2 (IKEv2) [22] is the standard protocol for establishing IPsec Security Associations. It operates over UDP and relies on a strictly defined exchange sequence.

A critical limitation of IKEv2 is how it handles packet sizes during the initial handshake (`IKE_SA_INIT`).

- **No Application-Layer Fragmentation:** The IKEv2 fragmentation mechanism (RFC 7383) only works for encrypted payloads *after* keys are established. It cannot fragment the initial `IKE_SA_INIT` message.
- **Reliance on IP Fragmentation:** If the `IKE_SA_INIT` packet exceeds the Path MTU (PMTU), it must rely on IP-level fragmentation.
- **The 1280-Byte Hard Limit:** To ensure reliability across the public Internet and avoid packet drops by middleboxes that block IP fragments, for maximal network routes compatibility, the entire UDP packet should fit within the minimum IPv6 MTU of **1280 B** [10].[2]

[2]For Ethernet networks, the PMTU usually achieves 1500 B, yet still relatively small compared to typical lattice KE payloads.

In order to figure out the protocol overheads and thus determining the maximum allowable size for a KEM public key or ciphertext, we have configured fixed algorithm suite in our experiment for both parties and captured their IPv6 packet during the SA establishment. The results are shown in Table 1, from which we observe a fixed overhead for IKE_SA_INIT, compared to pure Key Exchange Payload, by 208 and 241 Bytes respectively for the two directions. We precisely summarize the terms that account for these overheads in Table 2 (since we have fixed SA selection, size distinction only comes from Notify Payloads).

Standard lattice-based KEMs rely on conservative parameter choices to achieve negligible decryption failure rates, which leads to overly large public keys and ciphertexts that routinely exceed IPv6 packet constraints. Under the *mandatory IPv6 minimum MTU of 1280 bytes*—the global baseline required for universal deployment—even NIST-recommended ML-KEM-768 (Level 3) fails to fit without IP fragmentation, as its 1184-byte public key combined with IKEv2 overhead exceeds the safe payload limit. When relaxing to the *practical 1500-byte MTU* widely deployed in cloud and Ethernet networks, ML-KEM-1024 (Level 5) still cannot avoid fragmentation, as its key and ciphertext sizes remain well beyond the available budget. In both cases, standard ML-KEM is forced to rely on `IKE_INTERMEDIATE` (RFC 9242), which split key material across additional round trips and introduce nontrivial handshake latency and complexity. Remark that hybrid deployment remains compatible with our approach whenever the remaining MTU budget permits an additional classical DH share in IKE_SA_INIT.

3 AKE Security under Large DFR

This section establishes the cryptographic foundations for safely deploying KEMs with non-negligible decryption failure rates in AKE protocols. We identify two independent security requirements—IND-CPAF and Rigidity—and prove their sufficiency for IKEv2. The exposition proceeds in four steps:

1. **IND-CPA is insufficient** (§3.1–3.2): We abstract SIGMA AKE and give a counterexample where an IND-CPA KEM leaks the session key through observable failure behavior.

2. **IND-CPAF definition** (§3.3): Motivated by the counterexample, we formalize IND-CPAF, strengthening IND-CPA with a failure oracle and requiring pseudorandomness under all decapsulation outcomes.

Table 2: IKEv2 IKE_SA_INIT Overhead Breakdown (Bytes) Summarized from Captured Packets in Real Experiment.

Direction	Field	Size (B)
	I → R (Initiator to Responder)	
Common	SPI	16
	Next Payload	1
	Version	1
	Exchange Type	1
	Flags	1
	Msg ID	4
	Length	4
	SA	48
	KE Header	8
	Nonce	36
Notify	NAT_DETECTION_SOURCE_IP	28
	NAT_DETECTION_DESTINATION_IP	28
	IKEV2_FRAGMENTATION_SUPPORTED	8
	SIGNATURE_HASH_ALGORITHMS	16
	REDIRECT_SUPPORTED	8
	Total	**208**
	R → I (Responder to Initiator)	
Common	(Omitted)	...
Notify	NAT_DETECTION_SOURCE_IP	28
	NAT_DETECTION_DESTINATION_IP	28
	IKEV2_FRAGMENTATION_SUPPORTED	8
	SIGNATURE_HASH_ALGORITHMS	16
	Certificate Request	25
	CHILDLESS_IKEV2_SUPPORTED	8
	MULTIPLE_AUTH_SUPPORTED	8
	Total	**241**

3. **Rigidity is also necessary** (§3.4): We show IND-CPAF alone is insufficient: a non-rigid KEM silently outputting a predictable key enables MAC forgery and session compromise. We formalize Rigidity to mandate explicit error signaling.

4. **Sufficiency for IKEv2** (§3.5): We prove that a KEM satisfying

both IND-CPAF and Rigidity suffices for SK-security of the encrypted SIGMA (IKEv2) protocol in the Canetti-Krawczyk model.

3.1 A Generic Model of KEM-Based AKE

To analyze the security implications of KEM failures, we abstract the core logic of modern Authenticated Key Exchange (AKE) protocols, specifically those following the SIGMA design paradigm [23] (e.g., IKEv2, TLS 1.3). We focus on the interaction between the key exchange mechanism and the protocol state machine.

Consider a two-party protocol between an Initiator $\mathcal{I}$ and a Responder $\mathcal{R}$, each holding a long-term signing key pair $(\mathsf{pk}_{\mathcal{I}}, \mathsf{sk}_{\mathcal{I}})$ and $(\mathsf{pk}_{\mathcal{R}}, \mathsf{sk}_{\mathcal{R}})$. The 4-pass protocol can be divided into three logical phases:

1. **Key Exchange Phase:**
   - $\mathcal{I}$ generates an ephemeral KEM key pair $(pk, sk) \leftarrow \mathsf{KGen}$ and sends pk (pass 1).
   - $\mathcal{R}$ encapsulates a shared secret K against pk: $(K, ct) \leftarrow \mathsf{Encap}(pk)$. $\mathcal{R}$ sends ct (pass 2).
   - $\mathcal{I}$ decapsulates the ciphertext: $K' \leftarrow \mathsf{Decap}(sk, ct)$.
   - Note: If decapsulation fails, $K' \neq K$.
2. **Key Confirmation (Authentication) Phase:** The parties exchange authenticated messages that bind their identities to the key exchange transcript. This serves both to mutually authenticate and to confirm that they share the same K.
   - $\mathcal{I}$ computes a signature $\sigma_{\mathcal{I}} = \mathsf{Sign}(\mathsf{sk}_{\mathcal{I}}, (\text{“I”}, pk, ct))$ and a MAC $\tau_{\mathcal{I}} = \mathsf{MAC}(k_a, (\text{transcript}))$, and sends $(\mathsf{ID}_{\mathcal{I}}, \sigma_{\mathcal{I}}, \tau_{\mathcal{I}})$ (pass 3).
   - $\mathcal{I}$ first verifies the signature: if $\mathsf{Verify}(\mathsf{pk}_{\mathcal{I}}, (\text{“I”}, pk, ct), \sigma_{\mathcal{I}}) = 0$, $\mathcal{I}$ **aborts immediately**. Otherwise, $\mathcal{I}$ then verifies $\tau_{\mathcal{I}}$ using k'_a.
   - **Critical Branching:**
     - If $\mathsf{Verify}(k'_a, \tau_{\mathcal{I}}) = 1$, $\mathcal{I}$ accepts the session, transitions to state `ESTABLISHED`, and outputs ssk'.
     - If $\mathsf{Verify}(k'_a, \tau_{\mathcal{I}}) = 0$, $\mathcal{I}$ rejects the session, transitions to state `ABORT`, and erases ssk'.
   - $\mathcal{R}$ symmetrically sends its signed authentication message (pass 4). $\mathcal{I}$ verifies the signature first, then the MAC, and branches exactly as $\mathcal{R}$ does.

3. **Key Derivation Phase:** The parties derive separate keys for authentication and session encryption using a pseudorandom function (PRF). Note that this phase is sequential and there are two kinds of keys for derivation and acceptance, namely:

   - $\mathcal{I}$ computes $k'_a = \mathsf{PRF}(K', 0)$ after finishing the two-message exchange during the first phase, and use it in the second phase, i.e. authentication.
   - $\mathcal{R}$ computes both $k_a = \mathsf{PRF}(K, 0)$ and $ssk = \mathsf{PRF}(K, 1)$ after receiving the authentication message in the second phase. Then it use k_a in both verification of the received MAC, and the calculation of the response MAC; ready to use ssk to guarantee the established channel security.
   - $\mathcal{I}$ verifies the authentication response sent by $\mathcal{R}$ using k'_a. If verification is valid, it derives $ssk' = \mathsf{PRF}(K', 1)$ and ready to use ssk' to guarantee the established channel security.

The Observable Information Leakage Critically, the signatures bind the key exchange materials (pk, ct) to the peers' identities. Any active tampering with ct by a network adversary will cause signature verification to fail *before* the MAC is ever checked, resulting in an immediate abort that reveals nothing about the KEM decapsulation outcome. Consequently, the only scenario where MAC verification (and hence the KEM success/failure status) is reached is when the original, untampered ct is delivered. Under this condition, the transition to `ESTABLISHED` or `ABORT` after MAC verification becomes a *validity oracle* for the KEM ciphertext:

- **Acceptance** implies $k'_a = k_a$, which (with overwhelming MAC security) implies $K' = K$ (Success).[3]
- **Rejection** implies $k'_a \neq k_a$, which implies $K' \neq K$ (Failure).

While the adversary does not learn the session key ssk directly, they learn whether the underlying K was correct, which violates the isolation assumption of the standard IND-CPA model.

[3]This implication relies on the MAC key remaining pseudorandom. It can fail if decapsulation silently outputs $K' \neq K$ rather than $\perp$; we rule out such cases by rigidity in §3.4.

3.2 A Counterexample: IND-CPA is Insufficient

In this section, we provide a concrete mathematical counter-example to prove that IND-CPA security is insufficient for AKE protocols when decryption failures are observable. We demonstrate that a standard IND-CPA secure KEM, when combined with a standard secure PRF, can result in a protocol where the session key is trivially distinguishable from random, provided the adversary can observe decryption failures (e.g., via connection termination).

3.2.1 Construction of the Counter-Example

Let $F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^m$ be a secure PRF. Let $\mathcal{E} = (\mathsf{KGen}, \mathsf{Encap}, \mathsf{Decap})$ be an IND-CPA secure KEM. Using F, we construct a modified KEM $\mathcal{E}'$ which is identical to $\mathcal{E}$ except for the decapsulation algorithm Decap':

- $\mathsf{Decap}'(sk, ct)$:
  1. Compute the candidate shared secret: $K \leftarrow \mathsf{Decap}(sk, ct)$.
  2. Compute a "preview" of the session key: $y = F(K, 1)$ (w.l.o.g, assume the MSB of $F(0,1)$ is 0).
  3. Check the first bit (MSB) of y.
     - If $y[0] = 0$, return 0 as the final decryption result.
     - If $y[0] = 1$, return K (Success).

Validity of Components. It is easy to see that $\mathcal{E}'$ remains IND-CPA secure: Since the IND-CPA adversary is not allowed to query the decapsulation oracle on ct^*, nor does the game provide any feedback on whether Decap' would succeed or fail internally, the distribution of ct^* remains identical to the original scheme $\mathcal{E}$. Therefore, the IND-CPA advantage against $\mathcal{E}'$ is the same as against $\mathcal{E}$.

3.2.2 The Attack

We analyze the Generic AKE protocol (Section 3.1) instantiated with our modified KEM $\mathcal{E}'$ and the PRF F. Recall that the session key is derived as $ssk = F(K, 1)$. The attack shows how an adversary can learn information about the session key simply by observing whether the protocol succeeds or fails.

1. **Session Establishment:** The adversary $\mathcal{A}$ observes a session between $\mathcal{I}$ and $\mathcal{R}$.
2. **Observation:** $\mathcal{A}$ waits until the session is successfully `ESTABLISHED` (i.e., $\mathcal{I}$ successfully verifies the MAC and does not abort).
3. **Deduction:**
   - Acceptance implies that the MAC verification succeeded, meaning $\mathcal{I}$'s decapsulation via Decap' output the correct shared secret K rather than the invalid key 0.
   - By the artificial definition of $\mathcal{E}'$, Decap' only outputs the correct K if the first bit of the "previewed" session key $y = F(K, 1)$ is 1 (i.e., $y[0] = 1$).
   - Consequently, $\mathcal{A}$ learns that the first bit of the actual session key is 1 ($ssk[0] = 1$).
4. **Distinguishing:** $\mathcal{A}$ issues a Test query on the established session to obtain a challenge key ssk^*.
   - If $ssk^*[0] = 0$, $\mathcal{A}$ outputs **Random**.
   - If $ssk^*[0] = 1$, $\mathcal{A}$ outputs **Real**.

Advantage Calculation.

- **Case Real:** If ssk^* is the real session key, we know with probability 1 that $ssk^*[0] = 1$. Thus, $\mathcal{A}$ always guesses "Real".
- **Case Random:** If ssk^* is a random string from the key space, its first bit is 0 or 1 with equal probability.
  - If $ssk^*[0] = 0$ (Prob 0.5), $\mathcal{A}$ correctly guesses "Random".
  - If $ssk^*[0] = 1$ (Prob 0.5), $\mathcal{A}$ incorrectly guesses "Real".

The advantage of $\mathcal{A}$ is:

$$\mathbf{Adv}_{\mathcal{A}} = |\Pr[\mathcal{A} \to 1 \mid \text{Real}] - \Pr[\mathcal{A} \to 1 \mid \text{Random}]| = |1 - 0.5| = 0.5$$

This non-negligible advantage demonstrates that standard IND-CPA security of the underlying KEM is completely insufficient to guarantee AKE security when decryption failures are observable. The protocol's acceptance criteria create a side channel that leaks critical entropy of the derived session key.

3.2.3 Conclusion

This counter-example isolates and successfully attacks a precise flaw: an IND-CPA secure KEM can internally condition its success/failure behavior on the secret encapsulated material, and this behavior is made visible through the protocol's observable state. The standard IND-CPA model is blind to this channel—it does not grant the adversary a success/failure oracle—and is thus insufficient for upper-layer AKE security where such an oracle is naturally available. This motivates the IND-CPAF security notion introduced in Section 3.3, which explicitly grants the adversary access to a validity oracle and requires that the encapsulated secret remain pseudorandom even in its presence.

3.3 The IND-CPAF Security Notion

Motivated by the counterexample in Section 3.2, we introduce a refined security notion: **IND-CPAF** (Indistinguishability under Chosen Plaintext Attack with Failure). This definition is tailored for *ephemeral* KEM usage in protocols like IKEv2. It captures the adversary's ability to learn whether decapsulation succeeded via protocol feedback, and strictly requires that the session key remains secure regardless of the decapsulation outcome.

3.3.1 Formal Definition

As depicted in Figure 1, we define the IND-CPAF security game. Our IND-CPAF game differs from the standard IND-CPA game in one critical aspect:

1. **Failure Oracle:** The adversary can access a restricted oracle $\mathcal{O}_{\mathsf{FC}^*}(ct)$. This oracle returns a bit d indicating whether $\mathsf{Decap}(\mathsf{sk}, ct) = K_e$ where (K_e, ct) was obtained from $\mathsf{Encap}(\mathsf{pk})$ (i.e., whether decapsulation matches the encapsulation intent). Queries on any $ct \neq ct^*$ are rejected with $\perp$.

Definition 1 (IND-CPAF security, KEM)**.** We define the IND-CPAF security game as in Figure 1. The advantage of an adversary $\mathcal{A}$ against a KEM $\mathcal{E}$ is $\mathsf{Adv}^{\mathsf{IND\text{-}CPAF}}_{\mathcal{E},\mathcal{A}} = \left|\Pr[G^{\mathsf{IND\text{-}CPAF}}_{\mathcal{E},\mathcal{A}} = 1] - \frac{1}{2}\right|$. The scheme is IND-CPAF secure if this advantage is negligible in the security parameter λ.

Game $G^{\mathsf{IND\text{-}CPAF}}_{\mathsf{KEM},\mathcal{A}}$	Oracle $\mathcal{O}_{\mathsf{FC}^*}(ct)$
1 : $(\mathsf{pk}, \mathsf{sk}) \leftarrow\$ \mathsf{KGen}$	1 : **if** $ct \neq \mathsf{ct}^*$
2 : $(K_e, \mathsf{ct}^*) \leftarrow\$ \mathsf{Encap}(\mathsf{pk})$	2 : **return** $\perp$
3 : $b \leftarrow\$ \{0,1\}$	3 : **else**
4 : $K_0 = K_e,\ K_1 \leftarrow\$ \mathcal{K}$	4 : $K_d = \mathsf{Decap}(\mathsf{sk}, \mathsf{ct}^*)$
5 : $b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{FC}^*}}(\mathsf{pk}, \mathsf{ct}^*, K_b)$	5 : **return** $d = [\![K_d = K_e]\!]$
6 : **return** $[\![b' = b]\!]$	

Figure 1: IND-CPAF game for KEM.

3.3.2 Rationale and Justification

Failure oracle. We equip the adversary with a deterministic oracle $\mathcal{O}_{\mathsf{FC}^*}$ that, on input ct^*, returns a single bit indicating whether $\mathsf{Decap}(sk, ct^*)$ succeeds. This captures a restricted form of reaction attacks [18], abstracting protocol leakage to a one-bit success/failure signal. The oracle is strictly weaker than a decapsulation oracle: it reveals no key material and applies only to the challenge ciphertext. Repeated queries return the same bit and yield no additional information. Thus, IND-CPAF can be viewed as a minimal strengthening of IND-CPA under unavoidable one-bit leakage.

Restriction to ct^*. Queries are limited to ct^*, reflecting the structure of SIGMA-style protocols. The KEM-derived secret is used within authentication, so any modification of ct^* invalidates the corresponding authentication data and prevents the adversary from learning decapsulation validity. Hence, the only meaningful leakage concerns the original ct^*. This restriction relies on two assumptions: (i) the decapsulated secret is bound to authentication, and (ii) each (pk, ct^*) is used once and then erased (ephemeral setting).

3.3.3 Relationship to IND-CPA

By definition, IND-CPAF is strictly stronger than IND-CPA (IND-CPAF $\implies$ IND-CPA), as any IND-CPA adversary can be trivially simulated by simply ignoring the failure oracle. Conversely, IND-CPA does not imply IND-CPAF (IND-CPA $\not\Longrightarrow$ IND-CPAF), as demonstrated by the separation proof in Section 3.2. While the constructed scheme $\mathcal{E}'$ is IND-CPA secure, an IND-CPAF adversary can query $\mathcal{O}_{\mathsf{FC}^*}(ct^*)$ to learn the validity bit, which allows

it to distinguish the challenge key with probability 1 and trivially break IND-CPAF security.

3.4 The Necessity of Rigidity

Section 3.2 established that IND-CPA is insufficient, motivating IND-CPAF. However, IND-CPAF only guarantees pseudorandomness of the *correct* key K_e. It does not constrain the KEM's behavior when an internal failure occurs but is not explicitly signalled. One might think that the subsequent MAC-based key confirmation will always detect mismatches and abort, making such non-rigid failures harmless. We show that this intuition is false: *without rigidity, the standard unforgeability of the MAC is no longer sufficient to protect the session key, because the MAC key itself may no longer be pseudorandom.*

The security of a MAC (existential unforgeability) is guaranteed *only when its key is chosen uniformly at random and remains secret.* In an AKE protocol that uses a KEM, the MAC key is derived from the KEM output. When decapsulation fails silently—i.e., produces a value $K' \neq K_e$ without returning $\perp$—this faulty key may fall into a pathological subset of the key space for which the MAC behaves catastrophically. For example, one can construct a perfectly unforgeable MAC that, when instantiated with a key from a specific negligible-density set (e.g., keys ending with an all-zero suffix), completely exposes its key material or becomes trivially forgeable. In normal operation such keys never appear because they are negligibly probable under uniform sampling. However, a non-rigid KEM can *force* the MAC to be evaluated on exactly such a pathological key with non-negligible probability, thereby breaking the entire session.

In Appendix E we construct concrete primitives that are individually secure (IND-CPAF KEM, unforgeable MAC, secure PRF) but, when combined with a non-rigid KEM, enable an adversary to extract the initiator's session key. The construction leverages a MAC that reveals the base key when its second half is all-zero, and a KEM that occasionally outputs exactly such a key upon failure. Once the adversary learns the base key, it can forge the responder's MAC and derive the session key, completely compromising the initiator's session.

The root cause is that the KEM outputs a key (K') that is neither the correct K_e nor an explicit abort symbol $\perp$. To rule out such pathological interactions, we demand the KEM to be **rigid**: whenever it returns a non-$\perp$ value, that value must equal the correct encapsulated key K_e. Formally:

Definition 2 (Rigidity, KEM)**.** A KEM scheme $(\mathsf{KGen}, \mathsf{Encap}, \mathsf{Decap})$ is *ε-rigid* if

$$\mathop{\mathbb{E}}_{(\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{KGen}}\Big[\Pr\Big[K_d \neq \perp \wedge K_d \neq K_e \;\Big|\; \begin{matrix}(K_e, ct) \leftarrow \mathsf{Encap}(\mathsf{pk}) \\ K_d = \mathsf{Decap}(\mathsf{sk}, ct)\end{matrix}\Big]\Big] \leq \varepsilon.$$

Under rigidity, any decryption error results in $\perp$, protecting the protocol layer from ever using an unpredictable, potentially insecure key. Together with IND-CPAF, this ensures that every completed session derives its keys from a pseudorandom and correct shared secret. The full details of the counterexample are given in Appendix E.

3.5 Securing IKEv2 under IND-CPAF and Rigidity

We now show that the combination of IND-CPAF and rigidity is *sufficient* to prove the SK-security of the IKEv2 protocol. We stress that our analysis focuses on the 4-pass encrypted variant of the SIGMA protocol, denoted as Σ_1', which captures the core logic of IKEv2 (replacing DH with a general KEM).

We prove that the Σ_1' protocol satisfies SK-security in the Canetti-Krawczyk (CK) model, when instantiated with an ephemeral KEM that is both rigid and IND-CPAF secure. Our result shows that full IND-CCA security is not strictly necessary for IKEv2, provided the KEM satisfies these two tailored properties.

Protocol Flow Simplification We organize our proof sketch following Canetti and Krawczyk's IKEv2 analysis paper[6, 5]. For simplicity, they only presented the proof of Σ_1 as in Figure 2. Also note that, their proof in 2002 was only for the plaintext version, which did not provide rigorous analysis for the identity-protected/encrypted version (the actual adopted version of IKEv2). Therefore, we have incorporated a further analysis to cover the encryption for the last two messages. We use the notation $K\{\cdots\}$ to denote the AEAD encryption using the symmetric key K, presenting the analysis for the Σ_1' protocol in the figure.

Security Claim We first present our security claim, which corresponds to [5, Thm. 6] as Theorem 1, along with the definition for SK-security.

Definition 3 (SK-security [5])**.** An attacker with the above capabilities is called an SK-attacker. An AKE protocol π is called *SK-secure* if for all SK-attackers $\mathcal{A}$ running against π it holds:

The analyzed version Σ_1 in [6]:

$I \to R : s,\ g^x$
$R \to I : s,\ g^y$
$I \to R : s, IDi, \mathsf{Sign}_i(\text{"0"}, s, g^y, g^x), \mathsf{MAC}_{k_1}(\text{"0"}, s, IDi)$
$R \to I : s, IDr, \mathsf{Sign}_r(\text{"1"}, s, g^x, g^y), \mathsf{MAC}_{k_1}(\text{"1"}, s, IDr)$
where $ssk^* = \mathsf{PRF}_{g^{xy}}(0)$, $k_1 = \mathsf{PRF}_{g^{xy}}(1)$.

We will analyze Σ_1':

$I \to R : s,\ epk$
$R \to I : s,\ ect$, If $\mathsf{Decap}(ect) = \perp$, I abort or restart new handshake.
$I \to R : s, k_2\{IDi, \mathsf{Sign}_i(\text{"0"}, s, ect, epk), \mathsf{MAC}_{k_1}(\text{"0"}, s, IDi)\}$
$R \to I : s, k_2\{IDr, \mathsf{Sign}_r(\text{"1"}, s, epk, ect), \mathsf{MAC}_{k_1}(\text{"1"}, s, IDr)\}$
where $ssk^* = \mathsf{PRF}_{K(')}(0)$, $k_1 = \mathsf{PRF}_{K(')}(1)$, $k_2 = \mathsf{PRF}_{K(')}(2)$.

Figure 2: Simplified protocol flows for formal analysis.

P1) Any two completed matching sessions in a run of protocol π under attacker $\mathcal{A}$, given that neither of their owners has been corrupted, will always derive the same session key output, except for a negligible probability.

P2) $\mathcal{A}$ succeeds (in its test-session distinguishing attack) with probability not more than 1/2 plus a negligible fraction.

Note that the Canetti–Krawczyk (CK) model already captures *forward secrecy* through its description of the SK-attacker $\mathcal{A}$'s corruption ability before defining SK-security. Further details can be found in Appendix F .

Theorem 1. *Assuming the rigidity and IND-CPAF security of* KEM*, and the security of the underlying cryptographic functions* SIG*,* MAC*,* PRF*, the* Σ_1' *protocol is SK-secure in the post-specified model, as defined above.*

Our proof largely follows [5], but departs in several key aspects. Refer to Appendix G for the complete proof.

In SIGMA, MACs only prevent identity misbinding, while session-key consistency follows from signature unforgeability and transcript agreement. Under Diffie–Hellman, identical transcripts imply identical keys by deterministic derivation. This separation breaks with KEMs that have non-zero

decapsulation error: even when transcripts match, parties may derive different secrets. A natural approach is to let the protocol continue and rely on MACs to detect inconsistencies. However, this is unsound, as has been expained in Section 3.4. MAC security requires pseudorandom keys, while *KEM security under errors does not guarantee pseudorandomness of decapsulation outputs*. Consequently, it opens vulnerability for the protocol adversary to deceive honest parties into accepting insecure sessions. Thus, we instead let the decapsulator detect inconsistency and abort early. Then, this partitions the decapsulation output space and may leak information about valid secrets, as illustrated by our counterexample (Section 3.2). IND-CPAF and rigidity together can guarantee such protocol flow to smoothly prove secure.

Proof Sketch of P1 In short, guaranteeing message consistency suffices for Σ_1 and can be reduced to the unforgeability of the underlying signature scheme. In Σ_1', however, we further rely on KEM's rigidity. By definition, the attacker's advantage for obtaining $K_d \neq \perp \wedge K_d \neq K_e$ from an honest ciphertext is negligible.

Proof Sketch of P2 Our proof of Property P2 follows the proof strategy of [5] at a high level. The main difference lies in KEM's hardness assumption. Instead of reducing game hops (with respect to the guessed test session s_0) indistinguishability to constructing DDH distinguisher over tuples (g^x, g^y, z), we embed a KEM IND-CPAF challenge (pk, ct, K_b). Note that, we have to show such embedding is not recognizable by the adversary, otherwise the reduction fails. That means the guessed sessions, before or after being embedded with the challenge tuple, has to keep distributionally identical behaviors with respect to the corresponding game hops. The complexity that decryption failures introduce, as previously mentioned, is the fact that the MAC calculation or verification behavior (or additionally, the decryption of AEAD ciphertext under the encrypted SIGMA variant) can be hardly simulated without knowing whether the given challenge ciphertext correctly decrypts or not. However, since we have strengthened the requirement for KEM to provide such ability to answer whether the received ciphertext correctly decrypts, the protocol challenger can now make use of this information and smoothly embed the challenge while maintaining the protocol execution behavior unrecognizable by the adversary.

TRANSFORM $\mathcal{E}' = \mathsf{F}[\mathcal{E}, \mathsf{F}]$:	TRANSFORM $\mathcal{E}^\star = \mathsf{U}^\perp_m[\mathcal{E}', \mathsf{H}]$:
$\mathcal{E}'.\mathsf{KGen}'(1^\lambda)$:	$\mathsf{KGen}(1^\lambda)$:
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathcal{E}.\mathsf{KGen}(1^\lambda)$	$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathcal{E}'.\mathsf{KGen}'(1^\lambda)$
return $(\mathsf{pk}, \mathsf{sk})$	**return** $(\mathsf{pk}, \mathsf{sk})$
$\mathcal{E}'.\mathsf{Enc}'(\mathsf{pk}, m)$:	$\mathsf{Encap}(\mathsf{pk})$:
$c \leftarrow\!\$\ \mathcal{E}.\mathsf{Enc}(\mathsf{pk}, m)$	$m \leftarrow\!\$\ \mathcal{M}$
$t := \mathsf{F}(m, c)$	$\mathsf{ct} \leftarrow\!\$\ \mathcal{E}'.\mathsf{Enc}'(\mathsf{pk}, m)$
return $\mathsf{ct} = (c, t)$	$K = \mathsf{H}(m)$
$\mathcal{E}'.\mathsf{Dec}'(\mathsf{sk}, \mathsf{ct})$:	**return** (K, ct)
$Parse\,(c, t) \leftarrow \mathsf{ct}$	$\mathsf{Decap}(\mathsf{sk}, \mathsf{ct})$:
$m' = \mathcal{E}.\mathsf{Dec}(\mathsf{sk}, c)$	$m' = \mathcal{E}'.\mathsf{Dec}'(\mathsf{sk}, \mathsf{ct})$
if $m' = \perp$ **or** $\mathsf{F}(m', c) \neq t$	**if** $m' = \perp$
return $\perp$	**return** $\perp$
else return m'	**else**
	return $K' = \mathsf{H}(m')$

Figure 3: Split transform $\mathsf{R}[\cdot] = \mathsf{U}^\perp_m\big[\mathsf{F}[\cdot, \mathsf{F}], \mathsf{H}\big]$.

4 Rigid IND-CPAF KEM Generic Construction

In this section, we present a generic construction that transforms an IND-CPA-secure PKE scheme into an IND-CPAF-secure KEM. The construction preserves the correctness behavior of the underlying encryption scheme, including any non-negligible decryption failure rate. We prove that the resulting KEM satisfies IND-CPAF security under the same hardness assumptions as the original IND-CPA scheme. We also attach the discussion about FO-transform's availability in achieving IND-CPAF security, despite the fact that it is originally designed for obtaining perfectly correct IND-CCA KEMs. To begin with, our transform is described below.

Given an impefectly correct IND-CPA-secure PKE scheme $\mathcal{E} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ and two random oracles F and H, we can obtain an imperfectly correct IND-CPAF-secure KEM scheme $\mathcal{E}^\star = (\mathsf{KGen}, \mathsf{Encap}, \mathsf{Decap})$ by defining:

- $\mathsf{KGen}(\lambda)$: output $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\$\ \mathcal{E}.\mathsf{KGen}$.

- $\mathsf{Encap}(\mathsf{pk})$: sample $m \leftarrow\!\!\$\ \mathcal{M}$; encrypt $c = \mathsf{Enc}(\mathsf{pk}, m)$; compute tag $t = \mathsf{F}(m, c)$; set $\mathsf{ct} = (c, t)$ and derive $K = \mathsf{H}(m)$. Output (K, ct).
- $\mathsf{Decap}(\mathsf{sk}, \mathsf{ct})$: parse ct as (c, t); decrypt $m' = \mathsf{Dec}(\mathsf{sk}, c)$; if $m' = \perp$ or $\mathsf{F}(m', c) \neq t$, output nothing; otherwise output $K' = \mathsf{H}(m')$.

4.1 High-level Overview

As depicted in Figure 3, the complete transform above can be decomposed into two steps: $\mathsf{F}[\cdot]$ and $\mathsf{U}^{\perp}[\cdot, \mathsf{H}]$. This decomposition is only due to the convenience for modular proofs.

For the first step, we introduce a security notion called OW-CPAF to depict the security/property leverage of the middle PKE. At a high level, this step has guaranteed two things: (1) PKE absorbs most decryption failure cases, i.e. $m' \neq m$, into returning $m' = \perp$; (2) the resulting PKE makes it hard for the adversary to find a pre-image of the ciphertext ct^* (from decryption side) that is not $\perp$.

Secondly, given the middle PKE, we apply the explicit rejection and RO key derivation so that the resulting KEM achieves: (1) effectively spreads the entropy of the pre-key in the case of correct decryption; (2) for the failed or invalid decryption (judged by whether the decrypted pre-key $m' = \perp$), the decapsulation function outputs a failure signal $\perp$.

To present our proof, we first define OW-CPAF as a stepping stone in our complete proof. Its $\mathcal{O}_{\mathsf{FC}^*_m}$ oracle's function is similar to that of Definition 1, except the decryption failure is in PKE version. Formal definition is as follows:

Definition 4 (OW-CPAF, PKE)**.** The OW-CPAF (One-Wayness under Chosen Plaintext Attack with Failure) security of a PKE scheme is modeled through the OW-CPAF game, as illustrated in Figure 4. The advantage of an adversary $\mathcal{A}$ in the OW-CPAF game, denoted as $\mathsf{Adv}^{\mathsf{OW\text{-}CPAF}}_{\mathsf{PKE},\mathcal{A}}$, is defined as the probability of winning the game. We call a scheme OW-CPAF secure iff $\mathsf{Adv}^{\mathsf{OW\text{-}CPAF}}_{\mathsf{PKE},\mathcal{A}} = \Pr\left[G^{\mathsf{OW\text{-}CPAF}}_{\mathsf{PKE},\mathcal{A}} = 1\right]$ is negligible to the security parameter.

Definition 5 (Rigidity, PKE)**.** We define a public-key encryption scheme $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ to be ε-*rigid* if the advantage of any rigidity adversary $\mathcal{B}$ against PKE, in the rigidity game depicted in Figure 4, is at most ε. Namely, $\varepsilon \geq \mathsf{Adv}^{\mathsf{rigid}}_{\mathsf{PKE},\mathcal{B}} := \Pr\left[G^{\mathsf{rigid}}_{\mathsf{PKE},\mathcal{B}} = 1\right]$.

Note that, our definition for rigidity differs from that of [18], in which they only defined such property for deterministic PKEs, in a more limited

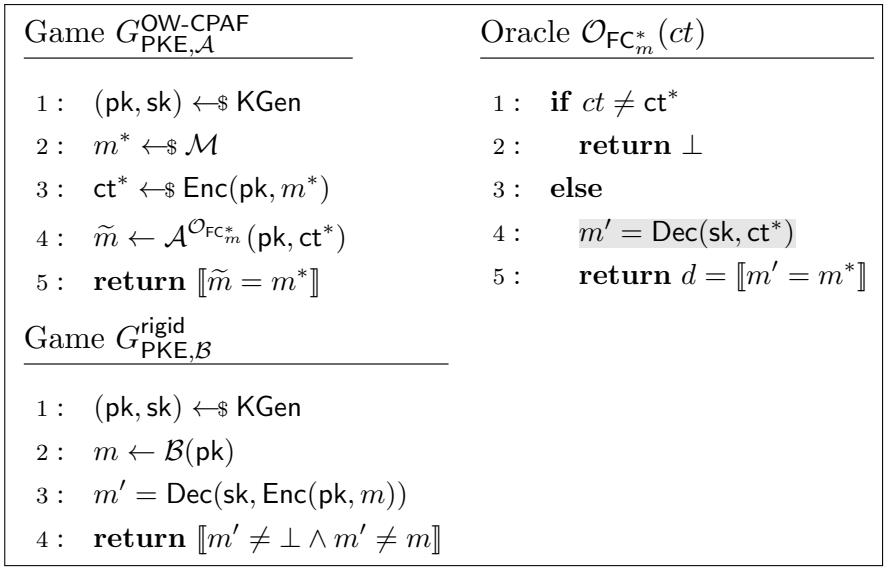


Figure 4: OW-CPAF game and Rigidity game for PKE.

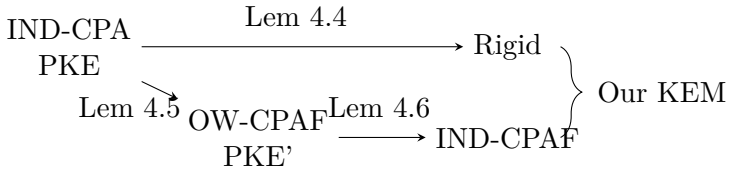


Figure 5: Proof for Thm 4.3.

way; while our definition applies for both probabilistic and deterministic PKE, introducing ε as a property parameter.

4.2 Security Claim

We first state the main result of this section as Theorem 2, whose proof is given by combining Lemma 1 to Lemma 3:

Theorem 2 (PKE IND-CPA $\overset{\text{ROM}}{\Rightarrow}$ KEM rigid + IND-CPAF)**.** *Assume the adversary issues at most q_{H}, q_{F} distinct queries respectively to the random oracle H and F, the output space size of F is denoted $|\mathcal{T}|$. Then,*

(1) KEM $\mathcal{E}^\star$ is ε-rigid in the random oracle model, where $\varepsilon := \frac{q_{\mathsf{F}}(q_{\mathsf{F}}-1)}{2|\mathcal{T}|}$ is the rigidity of the middle PKE $\mathcal{E}'$;

(2) If the base PKE $\mathcal{E}$ is δ-correct, then KEM $\mathcal{E}^\star$ is $(\delta+\varepsilon)$-correct in the random oracle model;

(3) For any IND-CPAF *adversary $\mathcal{A}$ against $\mathcal{E}^\star$, there exists an* IND-CPA *adversary $\mathcal{B}$ against $\mathcal{E}$, such that:*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPAF}}_{\mathcal{E}^\star,\mathcal{A}} \leq 4q_{\mathsf{H}} \cdot (1 - \frac{q_{\mathsf{F}}}{|\mathcal{T}|})^{-1} \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{E},\mathcal{B}}.$$

and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$.

The proof proceeds in a modular manner as depicted in Figure 5: first showing OW-CPAF security of $\mathcal{E}' = \mathsf{F}[\mathcal{E}, \mathsf{F}]$, and then IND-CPAF of $\mathcal{E}^\star = \mathsf{U}^\perp_m[\mathcal{E}', \mathsf{H}]$.

Lemma 1 ($\mathsf{PKE} \overset{\text{ROM}}{\Rightarrow} \mathsf{KEM\ rigid}$)**.** *For any rigidity adversary $\mathcal{D}$ against PKE scheme $\mathcal{E}'$ that issues at most q_{F} queries to the random oracle* F*, whose range space size is denoted $|\mathcal{T}|$, the advantage is bounded by: $\varepsilon := \mathsf{Adv}^{\mathsf{rigid}}_{\mathcal{E}',\mathcal{D}} \leq \frac{q_{\mathsf{F}}(q_{\mathsf{F}}-1)}{2|\mathcal{T}|}$. And the rigidity of the resulting KEM scheme $\mathcal{E}^\star$ is exactly the same as $\mathcal{E}'$.*

Proof. First, in the middle PKE $\mathcal{E}'$, any decrypted $m' = \mathsf{Dec}(\mathsf{sk}, c = \mathsf{Enc}(\mathsf{pk}, m))$ has to satisfy $\mathsf{F}(m', c) = \mathsf{F}(m, c)$ in order to return a non-abort value $m' \neq m$. Thus, the rigidity game's winning probability is bounded by the collision probability of the random oracle output F in total q_{F} times of queries. Using birthday bound, it gives $\approx \frac{q_{\mathsf{F}}(q_{\mathsf{F}}-1)}{2|\mathcal{T}|}$ when $q_{\mathsf{F}} \ll |\mathcal{T}|$.

Secondly, as long as $m' \neq \perp \wedge m' \neq m$ has been excluded, it naturally carries over to $K_d = \mathsf{Decap}(\mathsf{sk}, ct = \mathsf{Encap}(\mathsf{pk}))$, where $K_d = \mathsf{H}(m')$ and $K_e = \mathsf{H}(m)$. Namely, $K_d = K_e$ has been guaranteed whenever $m' \neq \perp$ (thus $K_d \neq \perp$). □

We leave the following lemmas' full proof to Appendix H and only give a high-level picture in the maintext.

Lemma 2 ($\mathsf{PKE\ IND\text{-}CPA} \overset{\text{ROM}}{\Rightarrow} \mathsf{PKE\ OW\text{-}CPAF}$)**.** *If PKE scheme $\mathcal{E}$ is δ-correct, then so is $\mathcal{E}'$. For any* OW-CPAF *adversary $\mathcal{A}$ that issues at most q_{F} queries to the random oracle* F*, there exists an* IND-CPA *adversary $\mathcal{B}$ such that:*

$$\mathsf{Adv}^{\mathsf{OW\text{-}CPAF}}_{\mathcal{E}',\mathcal{A}} \leq \left(1 - \frac{q_F}{|\mathcal{T}|}\right)^{-1} \cdot 4\ \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{E},\mathcal{B}}.$$

Proof Sketch. For the correctness of $\mathcal{E}'$, simply note that this transformation does not change the behavior of correct decryption; while for those

decryption returned with $m' \neq m$, Dec' "absorbs" most of them into abort case by checking whether $\mathsf{F}(m', c) = \mathsf{F}(m, c) = t$. Therefore, those ciphertexts that originally cause decryption failure will still fail, and the converse also still holds.

For the security reduction, the main idea is two-step: first, to simulate the failure oracle by simple 1/2 chance guessing; second, to utilize OW-CPAF adversary's ability to distinguish the IND-CPA challenge ciphertext c_b. We construct an IND-CPA adversary $\mathcal{B}$ against $\mathcal{E}$ as follows. $\mathcal{B}$ picks random m_0, m_1, receives $c_b^* = \mathcal{E}.\mathsf{Enc}(pk, m_b)$, and constructs two $\mathcal{E}'$ ciphertexts $\bar{c}_0 = c_b^* \| F(m_0, c_b^*)$ and $\bar{c}_1 = c_b^* \| F(m_1, c_b^*)$. It invokes the OW-CPAF adversary on both. With non-negligible probability(if $\mathsf{Adv}^{\mathsf{OW\text{-}CPAF}}_{\mathcal{E}',\mathcal{A}}$ is non-negligible), the adversary succeeds on the ciphertext whose tag matches the encrypted message (i.e., $\bar{c}_b$), while failing on the inconsistent one, as the random oracle makes it infeasible to recover the preimage from the tag. Hence $\mathcal{B}$ learns b by checking which message is recovered, and wins the IND-CPA game with advantage approximately equal to OW-CPAF adversary's advantage.

Lemma 3 (PKE $\mathsf{OW\text{-}CPAF}$ $\overset{\mathrm{ROM}}{\Rightarrow}$ KEM $\mathsf{IND\text{-}CPAF}$)**.** *If PKE $\mathcal{E}'$ is δ-correct, then KEM $\mathcal{E}^\star$ is $(\delta + \varepsilon)$-correct, where ε is the (negligible) rigidity bound for PKE $\mathcal{E}'$. For any* $\mathsf{IND\text{-}CPAF}$ *adversary $\mathcal{A}$ against $\mathcal{E}^\star$, issuing at most q_H queries to the random oracle* H*, there exists an* $\mathsf{OW\text{-}CPAF}$ *adversary $\widetilde{\mathcal{A}}$ against $\mathcal{E}'$ such that:* $\mathsf{Adv}^{\mathsf{IND\text{-}CPAF}}_{\mathcal{E}^\star,\mathcal{A}} \leq q_\mathsf{H} \cdot \mathsf{Adv}^{\mathsf{OW\text{-}CPAF}}_{\mathcal{E}',\widetilde{\mathcal{A}}}$

and the running time of $\widetilde{\mathcal{A}}$ is about that of $\mathcal{A}$.

Proof Sketch. The correctness bound is easy to verify: $K' \neq K$ whenever $m' = \perp$ or $m' \neq \perp \wedge m' \neq m$. For the security reduction, the main goal is to simulate the $\mathcal{O}_{\mathsf{FC}^*}$ oracle without the secret key or the decrypted value.

For the security reduction, since the failure oracle has already been provided by OW-CPAF challenge, the remaining part of proof essentially follows the same manner as in typical IND-to-OW reduction under RO model — which converts the IND adversary's advantage into RO pre-image search advantage, introducing a multiplicative loss factor of queried list size q_H.

4.3 IND-CPAF Secure Variant of MLKEM

Parameter Choice. We use exactly the same parameter sets as MLKEM [26], except for the compression parameters (d_t, d_u, d_v). The parameters for three security levels are changed from (12,10,4), (12,10,4), (12,11,4) to (8,8,4), (8,8,4), (9,8,3) respectively. Such parameter modification does not alter the

Table 3: Parameters, Sizes and Performance of MLKEM and eMLKEM. The column 'fails' records the actual failure times in total 100,000 tested iterations. Performance data are averaged over 100,000 iterations in 1000*cpucycles. Sec(C) shows the classical security of primal attack under core-SVP model. Compiled and tested on a Ubuntu 22.04 virtual machine with a 2.90GHz Intel(R) Core(TM) i7-10700 processor, reference-level baseline.

Scheme	Level	Sec.(C)	$\|\mathsf{pk}\|$	$\|\mathsf{ct}\|$	$\log \delta$	fails	KGen	Encap	Decap
MLKEM	1	115	800	768	-139	0	80	93	122
	3	182	1184	1088	-164	0	134	148	185
(IND-CCA)	5	255	1568	1568	-174	0	210	234	283
This work	1	115	544	672	-13.7	7	75	107	38
	3	182	**800**	**928**	-14.0	7	129	168	48
(IND-CPAF)	5	255	**1184**	**1152**	-12.3	24	197	245	60

underlying Module-LWE instance used for core-SVP estimates. The lattice hardness is entirely determined by the parameters governing key generation and error distributions (e.g., dimension, modulus, CBD parameters), while our changes only affect the compression parameters (d_t, d_u, d_v). Consequently, the underlying computational assumption remains *identical* to standard ML-KEM; only the correctness (decryption failure probability) changes.

Regarding the reduction loss, note that for ML-KEM/FIPS 203, concrete security levels are estimated from the hardness of the underlying lattice problem rather than from the tightness of the provable reduction. Our estimation follows exactly the same methodology, thus being a fair comparison.

Instantiation and Performance. For performance comparison shown in Table 3, we have modified the codes from the pq-crystals/kyber git repository[4] to our IND-CPAF version – eMLKEM. For symmetric primitives, eMLKEM follows the same specification as MLKEM, except we additionally instantiated the random oracle F using SHA3-256 HMAC. Note that, the decapsulation performance has leveraged to about 4-5x faster than MLKEM, due to the elimination of the re-encryption process needed by FO transform.

[4]https://github.com/pq-crystals/kyber (commit 4768bd3, Feb 16, 2025). Kyber is the original name for MLKEM.

Table 4: IPv6 IKE SA establishment latency (ms). Except for link (a) (10k/5k loops), all links iterated 1,000 loops. 'tll' and 'rtt' are from 'ping -6'. UF/DR/HF: UDP failure, decryption failure retry, handshake failure times.

Link	ttl	rtt	KEM	min	max	p99	med	avg	UF/DR/HF	KEM	min	max	p99	med	avg	UF/DR/HF
(a)	59	23.1	ecp384	31	171	76	50	51	0/0/0	ml1024	46	153	111	64	66	0/0/0
			eml768	34	88	71	48	49	**0/2/0**	eml1024	47	139	107	63	65	**0/2/0**
(b)	44	287.0	ecp384	577	2616	808	597	612	0/0/0	ml1024	586	2834	1746	796	780	0/0/0
			eml768	581	796	717	592	596	0/0/0	eml1024	568	1614	816	631	666	0/0/0
(c)	248	338.9	ecp384	662	8687	3672	674	867	0/0/0	ml1024	/	/	/	/	/	**unreachable**
			eml768	657	4677	3668	671	845	0/0/0	eml1024	669	16674	4684	683	1139	0/0/0
(d)	44	249.4	ecp384	483	16500	7501	494	1057	1/0/1	ml1024	453	22609	15460	461	1213	**32/0/32**
			eml768	450	18497	7493	490	870	2/0/2	eml1024	451	15455	3579	458	692	**1/1/1**

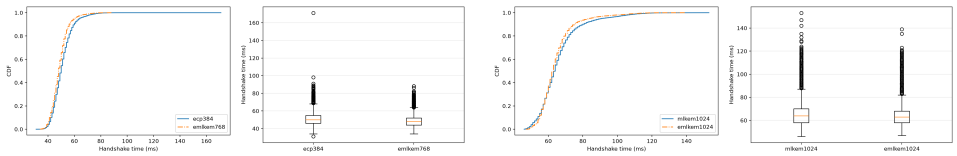
5 Experiment

At the time of writing (Mar. 2026), the strongSwan open-source library has not implemented the `IKE_INTERMEDIATE` exchange (RFC 9242 [30]), required by standard ML-KEM to split oversized payloads across multiple round trips. Therefore, we did not benchmark the full end-to-end latency difference between ML-KEM (which requires `IKE_INTERMEDIATE`) and our eMLKEM (which fits entirely in a single `IKE_SA_INIT` packet).

Instead, we conducted benchmarks under two controlled settings, both using only the standard `IKE_SA_INIT` flow: (1) **No fragmentation baseline**: We compare classical ECDH (ecp384) with our eMLKEM-768, both fitting within the 1280-byte IPv6 MTU and not triggering IP fragmentation; (2) **Fragmentation impact**: We compare standard ML-KEM-1024 with our eMLKEM-1024, where the former exceeds the 1500-byte Ethernet MTU and triggers IP layer fragmentation, while the latter remains unfragmented.

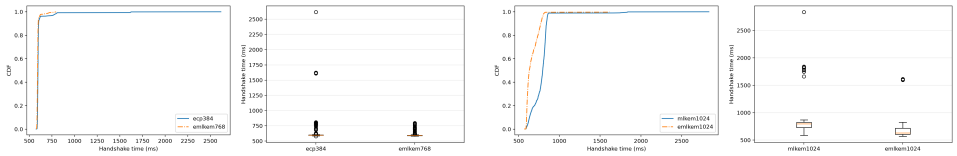
The additional latency overhead of `IKE_INTERMEDIATE` is well-understood and linear with network round-trip time (RTT), as it introduces at least one extra full round trip to the handshake. Thus, our benchmarking under these two settings fully captures the core performance advantages of our scheme: both the baseline efficiency comparable to classical cryptography and the dramatic robustness improvement under IP fragmentation.

We developed our KEM as a new plugin for strongSwan version 6.0.4, incorporating a limited retry mechanism to address decryption failures. The experimental evaluations were carried out on Alibaba Cloud Elastic Compute Service (ECS) instances, each configured with 2 vCPU, 4 GiB of memory, and a public IPv6 bandwidth of 20 Mbps, encompassing four intercontinental links. All tests utilized the same ECDSA certificates for authentication purposes. The further details regarding the experimental setup can



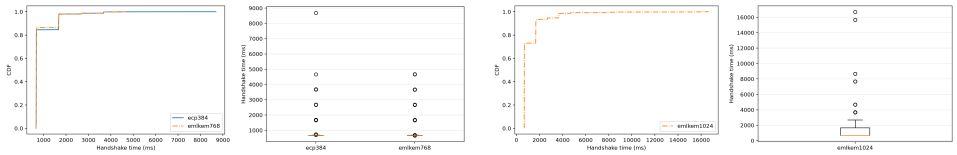
(a) BJ-SH

(a) BJ-SH



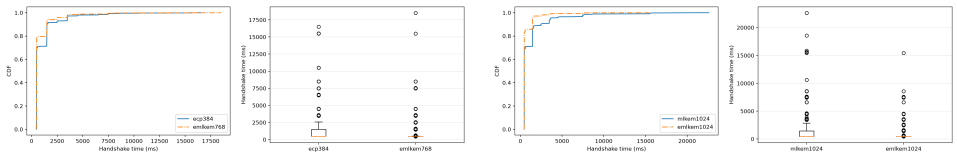
(b) BJ-KR

(b) BJ-KR



(c) BJ-SG

(c) BJ-SG



(d) UK-SH

(d) UK-SH

Figure 6: Comparing the latency between ECDH(ecp384) and eMLKEM-768 / MLKEM-1024 and eMLKEM-1024.

be found in Appendix C.

As summarized in Table 4, under the first setting (no IP fragmentation), the result shows the substitution of ECDH by eMLKEM has little impact on handshake behaviors. The latencies are still dominated by the network condition (one can check the distribution plots in Figure 6). Under the second setting, the result has validated the claim that IP layer fragmentation introduce extra transmission instability and obvious overheads, even causing some routes completely unreachable. For group (d), the handshake failure caused by failed UDP retransmission counts to 32 times in total 1,000 loops; whereas, eMLKEM-1024 that just fit in a single UDP packet under MTU has

maintained a relatively high stability, which only failed once among 1,000 loops.

Although one can see some decryption failure retries for group (a) among 10,000/5,000 loops, for group (d) among 1,000 loops, each retry has succeeded within `max_ike_init_retries` = 1; namely, no handshake failures were caused by the decryption failures.

6 Acknowledgements

Zidi, Yuyang, Long, and Zhenfeng were supported in part by the National Cryptologic Science Fund of China under Grant number 2025NCSF01007, the CAS Project for Young Scientists in Basic Research under Grant number YSBR-035, and ISCAS Basic Research Project under Grant number ISCAS-JCZD-202404. Qiang is supported in part by Google via the Digital Future Initiative, and research awards from the Stellar Development Foundation, and Input Output (iog.io) as part of the Cardano Vision 2026 research initiative.

References

[1] Erdem Alkim, Joppe W. Bos, Léo Ducas, Patrick Longa, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Chris Peikert, Ananth Raghunathan, and Douglas Stebila. 2020. FrodoKEM: Learning With Errors Key Encapsulation. Submission to the NIST Post-Quantum Cryptography Standardization Project. `https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/submissions/FrodoKEM-Round3.zip`

[2] Roberto Avanzi, J Bos, L Ducas, E Kiltz, T Lepoint, V Lyubashevsky, J Schanck, P Schwabe, G Seiler, and D Stehlé. 2020. CRYSTALS-Kyber: Algorithm Specifications And Supporting Documentation (version 3.0). *Technical report, National Institute of Standards and Technology.* `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`

[3] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2020. Crystals-kyber: a cca-secure module-lattice-based kem. In *IEEE EuroS&P*. IEEE, 353–381.

[4] Nina Bindel and John M Schanck. 2020. Decryption failure is more likely after success. In *International conference on post-quantum cryptography*. Springer, 206–225.

[5] Ran Canetti and Hugo Krawczyk. 2002. Security Analysis of IKE's Signature-based Key-Exchange Protocol (full version). Cryptology ePrint Archive, Paper 2002/120. https://eprint.iacr.org/2002/120

[6] Ran Canetti and Hugo Krawczyk. 2002. Security analysis of IKE's signature-based key-exchange protocol. In *CRYPTO*. Springer, 143–161.

[7] Cong Chen, Oussama Danba, Jeffrey Hoffstein, Andreas Hülsing, Joost Rijneveld, John M. Schanck, Peter Schwabe, William Whyte, Zhenfei Zhang, Tsunekazu Saito, Takashi Yamakawa, and Keita Xagawa. 2020. NTRU: Algorithm Specifications and Supporting Documentation (Version 3.0). Submission to the NIST Post-Quantum Cryptography Standardization Project. https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/submissions/NTRU-Round3.zip

[8] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. 2018. Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM. In *Progress in Cryptology - AFRICACRYPT 2018 - 10th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 7-9, 2018, Proceedings*. 282–305. doi:10.1007/978-3-319-89339-6_16

[9] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. 2020. SABER: Mod-LWR based KEM (Round 3 Submission). Submission to the NIST Post-Quantum Cryptography Standardization Project. https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/submissions/SABER-Round3.zip

[10] Dr. Steve E. Deering and Bob Hinden. 2017. Internet Protocol, Version 6 (IPv6) Specification. RFC 8200. doi:10.17487/RFC8200

[11] Jason A Donenfeld. 2017. WireGuard: Next Generation Kernel Network Tunnel.. In *NDSS*. 1–12.

[12] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. 2021. A cryptographic analysis of the TLS 1.3 handshake protocol. *Journal of Cryptology* 34, 4 (2021), 37.

[13] Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede. 2019. Decryption failure attacks on IND-CCA secure lattice-based schemes. In *Public-Key Cryptography–PKC 2019: 22nd IACR International Conference on Practice and Theory of Public-Key Cryptography, Beijing, China, April 14-17, 2019, Proceedings, Part II 22*. Springer, 565–598.

[14] Jan-Pieter D'Anvers, Mélissa Rossi, and Fernando Virdia. 2020. (One) failure is not an option: bootstrapping the search for failures in lattice-based encryption schemes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 3–33.

[15] Eiichiro Fujisaki and Tatsuaki Okamoto. 1999. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*. 537–554. doi:10.1007/3-540-48405-1_34

[16] Qian Guo, Thomas Johansson, and Alexander Nilsson. 2019. A generic attack on lattice-based schemes using decryption errors with application to ss-ntru-pke. *Cryptology ePrint Archive* (2019).

[17] Qian Guo, Thomas Johansson, and Jing Yang. 2019. A novel CCA attack using decryption errors against LAC. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 82–111.

[18] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. 2017. A Modular Analysis of the Fujisaki-Okamoto Transformation. In *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*. 341–371. doi:10.1007/978-3-319-70500-2_12

[19] Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. 2022. Failing gracefully: decryption failures and the Fujisaki-Okamoto transform. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 414–443.

[20] Loïs Huguenin-Dumittan and Serge Vaudenay. 2022. On IND-qCCA security in the ROM and its applications: CPA security is sufficient for TLS 1.3. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 613–642.

[21] Andreas Hülsing, Kai-Chun Ning, Peter Schwabe, Florian Weber, and Philip R Zimmermann. 2021. Post-quantum wireguard. In *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 304–321.

[22] Charlie Kaufman, Paul E. Hoffman, Yoav Nir, Pasi Eronen, and Tero Kivinen. 2014. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296. doi:10.17487/RFC7296

[23] Hugo Krawczyk. 2003. SIGMA: The 'SIGn-and-MAc'approach to authenticated Diffie-Hellman and its use in the IKE protocols. In *CRYPTO*. Springer, 400–425.

[24] Christian Majenz and Fabrizio Sisinni. 2024. Provable security against decryption failure attacks from LWE. In *Annual International Cryptology Conference*. Springer, 456–485.

[25] Daniele Micciancio and Oded Regev. 2007. Worst-case to average-case reductions based on Gaussian measures. *SIAM J. Comput.* 37, 1 (2007), 267–302.

[26] National Institute of Standards and Technology (US). 2024. *Module-lattice-based key-encapsulation mechanism standard*. Technical Report NIST FIPS 203. National Institute of Standards and Technology (U.S.), Washington, D.C. doi:10.6028/NIST.FIPS.203

[27] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. 2008. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology–CRYPTO 2008*. Springer, 554–571.

[28] Peter Schwabe, Douglas Stebila, and Thom Wiggers. 2020. Post-quantum TLS without handshake signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 1461–1480.

[29] Valery Smyslov. 2014. Internet Key Exchange Protocol Version 2 (IKEv2) Message Fragmentation. RFC 7383. doi:10.17487/RFC7383

[30] Valery Smyslov. 2022. Intermediate Exchange in the Internet Key Exchange Protocol Version 2 (IKEv2). RFC 9242. doi:10.17487/RFC9242

[31] Standards for Efficient Cryptography Group (SECG). 2010. *SEC 2: Recommended Elliptic Curve Domain Parameters*. Version 2.0. Certicom Research. `https://www.secg.org/sec2-v2.pdf`

A Open Science

Our implementation and testing codes for both standalone KEMs in Section 4.3 and integrated IPSec SA establishment experiment in Section 5 are available at: https://doi.org/10.5281/zenodo.22747226.

B Generative AI Usage

We utilized generative AI tools (specifically ChatGPT and Claude Code) to improve the clarity and presentation of the manuscript and to assist with experimental setup and coding. We emphasize that the **core conceptual ideas, theoretical analyses, and mathematical proofs** presented in this work are exclusively the original contributions of the authors.

All AI-generated content was rigorously reviewed, verified, and edited by the authors to ensure accuracy and alignment with our original research thoughts.

C Implementation Details

We describe the IKEv2 implementation, network conditions, and cryptographic parameters used in our experiments.

We implemented our KEM as a new plugin in strongSwan, forked from the master branch (commit 6856bfe, Mar. 4, 2026). The implementation is adapted from the existing ML-KEM plugin (ml) with modified parameter sets and registered as mlex, so that it can be configured during compilation using '–enable-mlex' flag and be specified with name 'emlkemXXX' in swanctl.conf, such as 'aes192-sha256-emlkem768'.

We also modified the IKE daemon (charon) to support limited retries upon 'ke_failed'. Specifically, we introduced a configuration parameter, `max_ike_init_retries`, and a retry counter in the IKE_SA_INIT state. Instead of immediately aborting on 'ke_failed', the initiator restarts the IKE_SA_INIT exchange until the retry limit is reached. In our experiments shown in Table 4, we have fixed `max_ike_init_retries` = 1. Consistent with our ephemeral-key assumption, strongSwan generates a fresh KEM

key pair for each new IKE_SA_INIT exchange (including each decryption-failure retry), while UDP retransmissions within the same handshake reuse the existing key pair.

Experiments were conducted on 5 Alibaba Cloud ECS instances of Ubuntu 22.04 system, each with 2vCPU 8GiB and 20 Mbps public IPv6 bandwidth. Among them, we selected 4 combinations of links, communicating via IPv6. Link (a) and (b) are through stable networks, while (c) and (d) are networks with high rtt or large packet loss rates. For each link, we compare the SA establishment latency between two pairs of algorithm suites: (1) aes192-sha256-ecp384 and aes192-sha256-emlkem768; (2) aes256-sha256-mlkem1024 and aes256-sha256-emlkem1024. All tests use the same pair of preset ECDSA (classical signature) certificates for authentication.

For timing method, since the default logging precision is one second, we augmented the logs with millisecond timestamps and measured handshake latency from "CREATED -> CONNECTING" to "CONNECTING -> ESTABLISHED", covering retry overhead whenever triggered.

D Primitives

Definition 6 (PKE)**.** A public-key encryption scheme PKE is a triple of probabilistic polynomial-time algorithm $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, called *key-generation*, *encryption*, and *decryption*, such that:

- KGen takes as input the security parameter 1^λ and outputs a pair of keys $(\mathsf{pk}, \mathsf{sk})$, in which pk is called the *public key* and sk is called the *secret key*. The public key pk defines a message space $\mathcal{M}$. We denote this by $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$, or simply write $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}$.
- Enc takes as input a public key pk and a message $m \in \mathcal{M}$, and outputs a *ciphertext* c. We denote this by $c \leftarrow\!\!\$\ \mathsf{Enc}(\mathsf{pk}, m)$.
- Dec takes as input a secret key sk and a ciphertext c, and outputs a message m or a special symbol $\perp$ denoting rejection. We denote this by $m = \mathsf{Dec}(\mathsf{sk}, c)$.

Definition 7 (KEM)**.** A Key encapsulation mechanism KEM is a triple of probabilistic polynomial-time algorithm $(\mathsf{KGen}, \mathsf{Encap}, \mathsf{Decap})$, called *key-generation*, *encapsulation*, and *decapsulation* respectively, such that:

- KGen takes as input the security parameter 1^λ and outputs a public-/secret-key pair $(\mathsf{pk}, \mathsf{sk})$. We denote this by $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$, or omit the security parameter and write $(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}$.

- Encap takes as input a public key pk. It outputs a *ciphertext* ct and an encapsulated value $K \in \{0,1\}^{\ell(\lambda)}$ where ℓ is the *key length*. We denote this by $(K, \mathsf{ct}) \leftarrow\!\!\$\ \mathsf{Encap}(\mathsf{pk})$
- Decap takes as input a secret key sk and a ciphertext ct, outputs a decapsulated value K or a special symbol $\perp$ denoting rejection. We denote this by $K = \mathsf{Decap}(\mathsf{sk}, \mathsf{ct})$.

Definition 8 (Correctness, PKE/KEM)**.** We call a public-key encryption scheme δ**-correct**, where the probability is taken over uniformly random m and r, and key pairs sampled from KGen:

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, c) \neq m \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen},\ m \leftarrow\!\!\$\ \mathcal{M},\ r \leftarrow\!\!\$\ \mathcal{R} \\ c = \mathsf{Enc}(\mathsf{pk}, m; r)\end{array}\right] \leq \delta$$

and we also refer to δ as the **failure probability**. Note that decryption rejection and decryption failure are distinct notions.
Similarly, δ-correctness can be defined for a key encapsulation mechanism as:

$$\Pr\left[\mathsf{Decap}(\mathsf{sk}, \mathsf{ct}) \neq K \;\middle|\; \begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KGen}, \\ (K, \mathsf{ct}) \leftarrow\!\!\$\ \mathsf{Encap}(\mathsf{pk})\end{array}\right] \leq \delta.$$

E Detailed Counterexample for the Necessity of Rigidity

We provide the full construction and attack sketched in Section 3.4. The building blocks are individual secure: a base IND-CPA KEM, a secure PRF, and a secure MAC. We then modify them to embed a "trapdoor" that only activates on keys with a specific structure (all-zero suffix), which occurs with negligible probability in normal usage.

E.1 Base Primitives

- $\mathsf{E} = (\mathsf{KGen}, \mathsf{Encap}, \mathsf{Decap})$: a perfectly correct IND-CPA secure KEM outputting shared secrets $K \in \{0,1\}^{2\lambda}$.
- $\mathsf{PRF} : \{0,1\}^{\lambda} \times \{0,1\}^{*} \to \{0,1\}^{\lambda}$: a secure pseudorandom function.
- $\mathsf{MAC} : \{0,1\}^{\lambda} \times \{0,1\}^{*} \to \{0,1\}^{\tau}$: a strongly unforgeable message authentication code.

E.2 Trapdoor Primitives

E.2.1 Extended MAC $\mathsf{M}^\star$

The key space is $\{0,1\}^{2\lambda}$. Write a key as $\kappa = \kappa_1 \| \kappa_2$ with $\kappa_1, \kappa_2 \in \{0,1\}^\lambda$.

$$\mathsf{M}^\star(\kappa_1 \| \kappa_2, m) = \begin{cases} \kappa_1 & \text{if } \kappa_2 = 0^\lambda, \\ \mathsf{MAC}(\kappa_1, m) & \text{otherwise.} \end{cases}$$

When κ is uniformly random, $\kappa_2 = 0^\lambda$ with probability $2^{-\lambda}$. Hence $\mathsf{M}^\star$ inherits the unforgeability of MAC: under a uniformly chosen key, the adversary never sees the trapdoor condition, and forging remains infeasible.

E.2.2 Extended PRF $\mathsf{F}^\star$

Similarly, define $\mathsf{F}^\star : \{0,1\}^{2\lambda} \times \{0,1\}^* \to \{0,1\}^\lambda$ as

$$\mathsf{F}^\star(\kappa_1 \| \kappa_2, m) = \mathsf{PRF}(\kappa_1, m)$$

For a uniformly random key, the output of $\mathsf{F}^\star$ is indistinguishable from a random function.

E.2.3 Non-rigid KEM $\mathsf{E}^\star$

We modify the base KEM to fail silently. Key generation and encapsulation are unchanged. Recall KEM E outputs $K \in \{0,1\}^{2\lambda}$. Decapsulation works as follows:

- Run $\mathsf{E.Decap}(sk, ct)$. If it returns $\perp$, output $\perp$.
- Otherwise split the decapsulated value returned by E as $K' = k_1 \| k_2$, where $k_1, k_2 \in \{0,1\}^\lambda$. With probability δ (chosen as non-negligible, e.g., 2^{-12}), output the **bad key** $K_{\mathsf{bad}} = k_1 \| 0^\lambda$. With probability $1-\delta$, output the **correct key** $K_{\mathsf{good}} = k_1 \| k_2$.

$\mathsf{E}^\star$ remains IND-CPA secure, because the IND-CPA game does not involve decapsulation. It is also IND-CPAF secure (the failure oracle merely gives the success/failure status, which helps negligibly). However, $\mathsf{E}^\star$ is **not rigid**:

$$\Pr[K_d \neq \perp \wedge K_d \neq K_e] = \delta.$$

E.3 Attack on the AKE Protocol

Now consider the Generic AKE protocol (Section 3.1) instantiated with $\mathsf{E}^\star$ as KEM, $\mathsf{M}^\star$ for MAC, and $\mathsf{F}^\star$ for session key derivation. Note that in the construction, the signature and MAC calculations are parallel. The MAC tag itself is not included in the signed content. Next, to maximize clarity, we let the MAC key k_a be the raw KEM output K (the attack equally applies when a PRF is used). The responder $\mathcal{R}$ uses K as MAC key and derives $ssk_{\mathcal{R}} = \mathsf{F}^\star(K, 1)$; the initiator $\mathcal{I}$ uses K' as MAC key and sets $ssk_{\mathcal{I}} = \mathsf{F}^\star(K', 1)$.

1. **Passive observation.** The adversary $\mathcal{A}$ observes an honest session, recording the initiator's ephemeral public key pk and the responder's ciphertext ct. It also intercepts $\mathcal{I}$'s signed authentication message (including the MAC tag $\tau_{\mathcal{I}}$).

2. **Decapsulation at $\mathcal{I}$.** $\mathcal{I}$ runs $K' \leftarrow \mathsf{Decap}^\star(sk, ct)$. With probability δ, $K' = k_1 \| 0^\lambda$ (trapdoor activated); with $1 - \delta$, $K' = k_1 \| k_2$.

3. **Leakage of k_1 via MAC.** $\mathcal{I}$ sends its signed authentication message, which includes the MAC tag $\tau_{\mathcal{I}} = \mathsf{M}^\star(K', \mathrm{trans}_{\mathcal{I}})$. When $K' = k_1 \| 0^\lambda$, the definition of $\mathsf{M}^\star$ yields $\tau_{\mathcal{I}} = k_1$. Such case occurs with non-negligible probability 2^{-12} from $\mathsf{E}^\star$'s construction, when $\mathcal{A}$ can simply take $\tau_{\mathcal{I}}$ as k_1.

4. **Impersonating $\mathcal{I}$.** $\mathcal{A}$ now sends to $\mathcal{R}$ the first authentication message, impersonating $\mathcal{I}$. It replays $\mathcal{I}$'s signature (which covers ("I", pk, ct)—all unchanged) and computes a forged MAC tag as $\tau_{\mathcal{A}} = \mathsf{MAC}(k_1, \mathrm{trans}_{\mathcal{I}})$. This is possible because $\mathcal{A}$ knows k_1.

5. **Acceptance at $\mathcal{R}$ and key recovery.** $\mathcal{R}$ verifies the signature (succeeds). It then verifies $\tau_{\mathcal{A}}$ using its local MAC key $K = k_1 \| k_2$: $\mathsf{M}^\star(k_1 \| k_2, \mathrm{trans}_{\mathcal{I}}) = \mathsf{MAC}(k_1, \mathrm{trans}_{\mathcal{I}}) = \tau_{\mathcal{A}}$. Now $\mathcal{R}$ is cheated into passing the verification and accept the derived session key $ssk_{\mathcal{R}} = \mathsf{F}^\star(K, 1) = \mathsf{PRF}(k_1, 1)$, which is totally known by the adversary.

E.4 Closing the Gap: A Modified Protocol View

The above exposes a subtle point: for $\mathcal{I}$ to accept, the adversary must send a tag that matches the constant k_1. But the adversary can only produce k_1 itself, not a function of k_1 under the original $\mathcal{R}$'s transcript. However, the adversary can alter the protocol flow slightly: it can relay the messages

in a way that the transcript seen by $\mathcal{I}$ matches the one used to compute the leaked k_1. In the SIGMA protocol, the transcript includes both parties' nonces and identities. If the adversary replays the exact same transcript as the original session, then the initiator will compute the same local MAC key and tag. Wait—the original session had a different binding (different nonces or session identifiers). The adversary cannot force a fresh $\mathcal{I}$ to accept replayed tags from a different session because the session identifiers differ. Thus the attack as described cannot succeed in the strict SIGMA model.

E.5 Why Rigidity is Still Mandatory

Despite the attack not leading to a full session compromise in the standard SIGMA setting, the constructed counterexample demonstrates a critical fragility: **the MAC alone cannot compensate for a non-rigid KEM if the MAC design itself relies on the key being pseudorandom**. The security proof of the AKE protocol reduces MAC unforgeability to the pseudorandomness of the KEM key. If the KEM can silently output a key from a negligible-density "bad" set, the proof collapses, because the MAC can no longer be assumed unforgeable under that key distribution. Thus, *rigidity is an essential requirement for the soundness of the cryptographic reduction, not necessarily for preventing a concrete attack.* In our IKEv2 security proof (Section 3.5), rigidity is used to ensure that whenever the KEM does not output $\perp$, the simulator can correctly decide whether to accept or reject without knowing the secret key, which would be impossible if the KEM could produce a wrong but non-$\perp$ key. The formal definition of rigidity (Definition 2) formalizes this requirement exactly.

F The Post-Specified Peer (PSP) Model

We adopt the *post-specified peer* (PSP) model as defined by Canetti and Krawczyk [6] to formalize the security of the KEM-based IKEv2 instantiation. This model is particularly suited for IKEv2-like protocols where the peer's identity is disclosed only after the initial exchange.

F.1 Sessions and Identifiers.

A party $P \in \mathcal{U}$ can initiate multiple protocol instances, each referred to as a *session*. To avoid ambiguity between local management and protocol-level identity binding, we distinguish between the following:

- **(Local) Session ID:** A unique 's' generated locally by party P to identify a session within its own internal state (e.g., the SPIs in IKEv2).
- **Session Triple** (P, s, Q)**:** The logical output of a completed session. It binds the local identity P and the local s to a specific peer Q. In the PSP setting, Q is initially set to a null value ($*$) and is only "post-specified" as a specific identity $Q \in \mathcal{U}$ upon the session's completion.

Following Section 2.1 of [6], in the PSP setting, a key exchange session is activated at a party P with parameters (P, s), where P is the local identity, s is a unique session identifier. When a session completes, it outputs a session key κ and a completion triple (P, s, Q), where Q is the identity of the party with whom P believes it has exchanged the key.

F.2 Session Stage and Internal State.

Each session belongs to a $\mathsf{stage} \in \{\textit{incomplete}, \textit{completed}, \textit{expired}\}$ and maintains its own internal states:

- **Incomplete:** The session is ongoing. P holds an ephemeral internal state state_s.
- **Completed (Accepted):** The session derives a session key κ and outputs the triple (P, s, Q). The internal state state_s is erased.
- **Expired:** The session key κ is erased, though the triple (P, s, Q) remains for definition purposes.

In the context of IKEv2, the internal state involves the algorithm randomness and secret exponent of the Diffie-Hellman exchange; and under the usage of post-quantum KEM, it also captures the KEM randomness and the ephemeral decapsulation key sk.

F.3 Matching Sessions.

The notion of matching sessions is central to defining freshness and authentication. According to Definition 2 in [6], two sessions (P, s) and (Q, s) are called **matching** if they share the same unique session identifier s, and one of the following conditions holds:

1. Session (Q, s) is *incomplete* (i.e., has not yet output a completion triple); OR
2. Session (Q, s) has *completed* and output the triple (Q, s, P).

This definition ensures that even if a peer has not yet finished its protocol execution, it can still be recognized as the matching counterpart to a completed session.

F.4 Adversarial Queries.

The adversary $\mathcal{A}$ is a PPT algorithm that controls the communication network. Following the PSP framework, $\mathcal{A}$ interacts with parties through the following queries:

- $\mathsf{Send}((P, s), m)$: Delivers message m to session s at party P.
- $\mathsf{Reveal\text{-}session\text{-}key}(P, s)$: Returns κ for a *completed* session.
- $\mathsf{Reveal\text{-}state}(P, s)$: Returns the internal ephemeral state of an *incomplete* session.
- $\mathsf{Corrupt}(P)$: Returns the long-term secret signing key of party P.
- $\mathsf{Test}(P, s)$: This query can only be issued to a *completed*, *fresh* session. If so, the challenger returns either κ or a random string based on a hidden bit $b \leftarrow\!\!\$\ \{0, 1\}$.

F.5 SK-Security and Freshness.

We restate the definition of SK-security and freshness as follows:

Definition 9 (SK-security [6])**.** An attacker with the above capabilities is called an SK-attacker. An AKE protocol π is called *SK-secure* if for all SK-attackers $\mathcal{A}$ running against π it holds:

P1) Any two completed matching sessions in a run of protocol π under attacker $\mathcal{A}$, given that neither of their owners has been corrupted, will always derive the same session key output, except for a negligible probability.

P2) $\mathcal{A}$ succeeds (in its test-session distinguishing attack) with probability not more than $1/2$ plus a negligible fraction.

Definition 10 (Freshness)**.** A session (P, s) that completed with triple (P, s, Q) is *fresh* if:

1. From the view of party P:
    - *incomplete* stage: no $\mathsf{Reveal\text{-}state}$ has been issued to (P, s);

- *completed* stage: no Reveal-session-key has been issued to (P, s, Q).

2. If a matching session for (P, s, Q) exists, which is (Q, s) or (Q, s, P), it must also satisfy from the view of party Q:

   - *incomplete* stage: no Reveal-state has been issued to (Q, s);
   - *completed* stage: no Reveal-session-key has been issued to (Q, s, P);
   - Under either *incomplete* or *completed* stage, no Corrupt has been issued to party Q.

F.6 Forward Secrecy.

In this model, *forward secrecy* is captured by the resilience of session keys to the corruption of long-term keys after the session has expired. Specifically, consider a session (P, s, Q) that has entered the *expired* status (at which point the session key κ is erased from P's memory) and its matching session (Q, s) at a party Q. The model requires that even if an adversary $\mathcal{A}$ issues a Corrupt(P) query after (P, s, Q) has expired, the session key κ of the matching session (Q, s) – provided it remains *fresh* – remains indistinguishable from a random string. This ensures that the compromise of P's long-term signing key does not affect the security of previously established and now expired session keys.

G Omitted Proofs for AKE Security

G.1 Proof for Plaintext Version of Theorem 1

Note that, what the original paper [5] has proven is actually a plaintext version of SIGMA protocol or IKEv2; namely, the identity protection is not provided (or the last two messages are not encrypted). Such variant actually requires less security guarantees for the underlying KEM (or DH in classical settings). As a warm-up, we first present the plaintext variant's security proof, which closely follows the original proof such that the distinctions can be easily compared.

We adapt and adjust their proof to accommodate our modifications. The key differences between our proof and the original proof include: (1) proof of the first property in Definition 9 (Section 4.2 of [5]); (2) proof of the second property: we modify and re-prove Lemma 7, Lemma 8.

Proof of P1

Plaintext version Σ_1'':

$I \to R : s,\ epk$

$R \to I : s,\ ect,$ If $\mathsf{Decap}(ect) = \perp$, I abort or restart new handshake.

$I \to R : s, IDi, \mathsf{Sign}_i(\text{"0"}, s, ect, epk), \mathsf{MAC}_{k_1}(\text{"0"}, s, IDi)$

$R \to I : s, IDr, \mathsf{Sign}_r(\text{"1"}, s, epk, ect), \mathsf{MAC}_{k_1}(\text{"1"}, s, IDr)$

where $ssk^* = \mathsf{PRF}_{K_e}(0)$, $k_1 = \mathsf{PRF}_{K_e}(1)$.

In the original proof of Property P1 for Σ_1, the key argument is that, upon acceptance, each party receives exactly the Diffie–Hellman exponent sent by the other. Determinism of the Diffie–Hellman exponentiation then guarantees that both parties derive the same shared secret g^{xy}. However, in our setting, consistency of the exchanged public values (namely, epk and ect in Σ_1'') alone does not suffice. In the presence of decryption failures, the initiator's shared secret $K' = \mathsf{Decap}(esk, ect)$ and the responder's shared secret K obtained from $(K, ect) \leftarrow\!\$\ \mathsf{Encap}(epk)$ may differ with probability δ. Consequently, we must additionally show that, upon acceptance, the event $K' \neq K$ occurs only with negligible probability.

Guaranteeing message consistency suffices for Σ_1 and can be reduced to the unforgeability of the underlying signature scheme. In Σ_1'', however, we additionally require the rigidity of the underlying KEM. By definition, any attacker's advantage for generating $K_d \neq \perp \wedge K_d \neq K_e$ is negligible, which concludes the proof.

Proof of P2 The major difference is introduced to Lemma 7 and Lemma 8 in [5], which we modify into Lemma 4 and Lemma 5 below, respectively.

Following the proof strategy in [5], we first define a simulator $\mathcal{S}$ that is capable of simulating the protocol Σ_1'' for the adversary $\mathcal{A}$. If at any point $\mathcal{A}$ corrupts a party, $\mathcal{S}$ hands out to $\mathcal{A}$ all the internal information of that party (including private signature key, session state for incomplete sessions, and session keys for unexpired sessions) and $\mathcal{S}$ stops operating that party (which is now under full control of $\mathcal{A}$). Upon a Reveal-state query against a specific (incomplete) session, $\mathcal{S}$ provides $\mathcal{A}$ with the internal state information for that session; similarly, if $\mathcal{A}$ performs a Reveal-session-key query against a (complete and unexpired) session then $\mathcal{S}$ provides $\mathcal{A}$ with the corresponding secret key output by that session. Note that at any point in its run $\mathcal{S}$ has full information to answer all of $\mathcal{A}$'s queries or perform the protocol actions on behalf of the uncorrupted parties. When $\mathcal{A}$ chooses a test session and performs its test query, $\mathcal{S}$ responds with the value of the session key

as output by the test session. When $\mathcal{A}$ stops, $\mathcal{S}$ stops too with the same output (0 or 1) as $\mathcal{A}$.

We introduce several variants of the above simulator $\mathcal{S}$ which by now we generically denote by $\hat{\mathcal{S}}$. An $\hat{\mathcal{S}}$ simulator is similar to $\mathcal{S}$ except for some differences:

1. Let N_s be an a-priori upper bound on the number of sessions that $\mathcal{A}$ initiates (i.e., sessions for which $\mathcal{A}$ issues an initiation activation upon which a party outputs a start message) during its run with and N_p parties. At the beginning of its run $\hat{\mathcal{S}}$ chooses the following values: a guessed number t chosen uniformly between 1 and N_s, an identity R_0 randomly chosen among the identities of the N_p parties in the protocol, a KEM challenge tuple (pk, ct^*, K_b), and protocol-derived keys, ssk^* used as session key, and k_1 used for MAC authentication.

2. Let the t-th session, i.e. the guessed session, initiated by $\mathcal{A}$ be (I_0, s_0). The following actions take place for *this* session with all values given in step 1, as long as an abort event does not happen. The start message of session (I_0, s_0) is generated by $\hat{\mathcal{S}}$ using pk. In case that session (R_0, s_0) is activated by $\mathcal{A}$ with R_0 as responder then $\hat{\mathcal{S}}$ outputs a response message on behalf of R_0 using ct^*. Also the MAC computation for respond messages uses the key k_1 chosen by $\hat{\mathcal{S}}$. If any of the sessions (I_0, s_0) or (R_0, s_0) complete then the secret session key is set to ssk^* as chosen by $\hat{\mathcal{S}}$.

3.The specification of the ways in which ssk^* and k_1 are chosen will determine the different variants of simulators $\hat{\mathcal{S}}$ as following:

$$
\begin{array}{lll}
\hat{\mathcal{S}}\text{-REAL}: & ssk^* \leftarrow \mathsf{PRF}_{K'}(0), & k_1 \leftarrow \mathsf{PRF}_{K'}(1) \\
\hat{\mathcal{S}}\text{-RPRF}: & ssk^* \leftarrow \mathsf{PRF}_{k}(0), & k_1 \leftarrow \mathsf{PRF}_{k}(1), \quad k \leftarrow\!\!\$\, \mathcal{K} \\
\hat{\mathcal{S}}\text{-ALLR}: & ssk^* \leftarrow\!\!\$\, \mathcal{K}, & k_1 \leftarrow\!\!\$\, \mathcal{K} \\
\hat{\mathcal{S}}\text{-HYBR}: & ssk^* \leftarrow\!\!\$\, \mathcal{K}, & k_1 \leftarrow \mathsf{PRF}_{k}(1), \quad k \leftarrow\!\!\$\, \mathcal{K} \\
\hat{\mathcal{S}}\text{-RAND}: & ssk^* \leftarrow\!\!\$\, \mathcal{K}, & k_1 \leftarrow \mathsf{PRF}_{K'}(1)
\end{array}
$$

where K' represents the encapsulated secret K_e (resp. the decapsulated secret K_d) for the responder (resp. the initiator) side; ssk^* represents the tested session key; k_1 is the MAC key in the 3rd or 4th message in Σ_1''.

4. If the t-th session is not the tested session selected by $\mathcal{A}$, or $\mathcal{A}$ does not issue a Test query, then $\hat{\mathcal{S}}$ aborts. Otherwise when $\mathcal{A}$ stops, $\hat{\mathcal{S}}$ stops too with the same output (0 or 1) as $\mathcal{A}$.

We denote by Guess the event that the t-th session guessed by simulator $\hat{\mathcal{S}}$ is exactly the tested session selected by $\mathcal{A}$. The following lemma is concerned with the actions of $\mathcal{A}$ and not directly with the behavior of

the above simulators. However, this lemma will be instrumental in claiming that under a GUESS event the above simulators do not abort their run. In fact, according to the definition in [5], $\hat{\mathcal{S}}$ also aborts if the property of any item in Lemma 4 does not hold, whereas the GUESS event guarantees that all these properties hold.

Lemma 4. *For all Σ_1''-attackers $\mathcal{A}$, the following holds except for negligible probability:*
A. Consider a regular run by $\mathcal{A}$ in which $\mathcal{A}$ chooses a tested session (i.e. accepted) with output (P, s, Q) where P is the initiator. Then:

(1) P and Q are never corrupted before expiration of the test session;

(2) Sessions (P, s) and (Q, s) are never revealed by $\mathcal{A}$;

(3) (Q, s) is initiated as responder with the start message sent by (P, s);

(4) (P, s) receives a response message after (Q, s) was activated as responder, and this message carries the same ect as in the response message output by (Q, s);

(5) Session (P, s) does not abort;

(6) The agreed en-/de-capsulated value $K_d = K_e$ are pseudorandom from adversaries' view.

B. Consider a regular run by $\mathcal{A}$ in which $\mathcal{A}$ chooses a tested session with output (Q, s, P) where Q is the responder. Then[5]*:*

(1) P and Q are never corrupted before expiration of the test session;

(2) Sessions (P, s) and (Q, s) are never revealed by $\mathcal{A}$;

(3) (P, s) has been initiated as initiator who has sent the start message to (Q, s);

(4) (Q, s) receives a response message after (P, s) was activated as initiator, and this message carries the same epk as in the start message output by (P, s);

(5) Session (Q, s) does not abort;

(6) The agreed en-/de-capsulated value $K_d = K_e$ are pseudorandom from adversaries' view.

[5]We have fixed some typos in the item 3-5 from [5].

Proof. For the proof of Part A, items (1), (3), and (5) are guaranteed by the definition of the tested session. Items (3) and (4) holds due to the unforgeability of the underlying signature scheme. The main difference from [5] is that, we add the item (6) in both cases. To prove (6), we rely on both the rigidity and the IND-CPAF security of the KEM. By definition, any attacker's advantage for

i) obtaining an honest ciphertext that yields $K_d \neq \perp \wedge K_d \neq K_e$;

ii) distinguishing between K_e and a uniformly random key,

are both negligible. Property (i) is guaranteed by the *rigidity* of the KEM, while property (ii) is guaranteed by the *IND-CPAF security* of the KEM.

The proof of Part B follows analogously to that of Part A.

□

Since different variants of simulator $\hat{\mathcal{S}}$ differ only in the t-th session, and since $\hat{\mathcal{S}}$ aborts rather than completing the t-th session when the GUESS event does not occur, $\mathcal{A}$ can only have a potential advantage in distinguishing different variants of $\hat{\mathcal{S}}$ when the GUESS event occurs. Furthermore, conditioned on the GUESS event, the advantage of $\mathcal{A}$ in distinguishing $\hat{\mathcal{S}}$-$\mathsf{RAND}(\mathcal{A})$ from $\hat{\mathcal{S}}$-$\mathsf{HYBR}(\mathcal{A})$, as well as in distinguishing $\hat{\mathcal{S}}$-$\mathsf{REAL}(\mathcal{A})$ from $\hat{\mathcal{S}}$-$\mathsf{RPRF}(\mathcal{A})$, can be bounded by embedding the CPAF challenge of the KEM into the tested session. The advantage of $\mathcal{A}$ in distinguishing $\hat{\mathcal{S}}$-$\mathsf{ALLR}(\mathcal{A})$ from $\hat{\mathcal{S}}$-$\mathsf{HYBR}(\mathcal{A})$, as well as in distinguishing $\hat{\mathcal{S}}$-$\mathsf{RPRF}(\mathcal{A})$ from $\hat{\mathcal{S}}$-$\mathsf{ALLR}(\mathcal{A})$, can be bounded by exploiting the properties of the PRF.

Lemma 5. *For all Σ_1''-attackers $\mathcal{A}$, $\hat{\mathcal{S}}$-$\mathsf{RAND}(\mathcal{A}) \approx \hat{\mathcal{S}}$-$\mathsf{HYBR}(\mathcal{A})$, relying on the IND-CPAF security of the underlying KEM.*

Proof. $\hat{\mathcal{S}}$-RAND and $\hat{\mathcal{S}}$-HYBR differ only in that, whether the PRF key for deriving MAC key k_1 is equal to the IND-CPAF challenge $K_0 = K_e$, or a independently sampled random string k.

The adversary $\mathcal{A}$'s distinguishing advantage between these simulators, if there is any, can be used to construct a distinguisher $\mathcal{D}$ to break the IND-CPAF security of KEM by embedding the challenge $(\mathsf{pk}, \mathsf{ct}^*, K_b)$ into the execution of the tested session and its peer. This embedding follows almost in the same way as has been described in [5] for embedding DDH challenge (g^x, g^y, z). In addition to replacing these elements in their corresponding order, the only difference lies in the MAC verification step, where the protocol challenger (the simulator) has to query the oracle $\mathcal{O}_{\mathsf{FC}^*}$ provided by IND-CPAF challenge in order to decide whether to accept the MAC or not (since

it doesn't know whether the decapsulated K_0 matches the encapsulated K, which are used for MAC calculation and verification respectively). The complete process of embedding is described as follows, we assume (I_0, s_0) and (R_0, s_0) are the guessed test session and its peer:

1. $\mathcal{D}$ replaces epk in the start message of (I_0, s_0) with pk from the IND-CPAF challenge, i.e. setting $[s_0, \mathsf{pk}]$ as the start message from (I_0, s_0).

2. If (R_0, s_0) receives a response message involving pk then $\mathcal{D}$ replaces ect in the response message with ct^* from the IND-CPAF challenge. That is, $\mathcal{D}$ produces $[s_0, \mathsf{ct}^*]$ as the response message from (R_0, s_0). Otherwise, if (R_0, s_0) receives a response message with epk different than pk then $\mathcal{D}$ conduct the general behavior, i.e. generating ect from epk and continues.

3. Upon (I_0, s_0) receiving a response message containing ect, $\mathcal{D}$ queries $\mathcal{O}_{\mathsf{FC}^*}(ect)$ to obtain the value d. If $d = 0/\perp$, $\mathcal{D}$ aborts; If $d = 1$, then $\mathcal{D}$ generates a response message from (I_0, s_0) in the form $[s_0, I_0, \mathsf{SIG}_{I_0}(\text{"0"}, s_0, \mathsf{ct}^*, \mathsf{pk}), \mathsf{MAC}_{k_1}(\text{"0"}, s_0, I_0)]$, where $k_1 = \mathsf{PRF}_{K_b}(1)$, and K_b is the third element in the challenge.

4. If (R_0, s_0) receives a response message, R_0 verifies the signature included in it under the identity public key of the initiator (as it appears in the response message). Then R_0 verifies the MAC through recalculation using $k_1 = \mathsf{PRF}_{K_b}(1)$, derived from K_b. If all verifications are valid, $\mathcal{D}$ then makes (R_0, s_0) output a finish message of the form:

   $[s_0, R_0, \mathsf{SIG}_{R_0}(\text{"1"}, s_0, \mathsf{pk}, \mathsf{ct}^*), \mathsf{MAC}_{k_1}(\text{"1"}, s_0, R_0)]$; and R_0 accepts the random session key ssk^* (as in $\hat{\mathcal{S}}$-RAND), ready to be tested.

5. On incoming finish message to (I_0, s_0), I_0 verifies the signature in it under the identity public key of the responder. Then, the MAC's verification is conducted in a normal manner for I_0, i.e. recalculating MAC from $k_1 = \mathsf{PRF}_{K_b}(1)$ and comparing it to the received value. If all verifications are valid, I_0 accepts the random session key ssk^* (as in $\hat{\mathcal{S}}$-RAND), ready to be tested.

Now we need to show that if the embedded $K_b = K_0$, the game is identical to $\hat{\mathcal{S}}$-RAND, and $K_b = K_1$ identical to $\hat{\mathcal{S}}$-HYBR, from the adversary $\mathcal{A}$'s view. First, the behaviors of all other sessions in the simulators, except for the guessed test sessions, remain exactly the same as not whether or not the challenge has been embedded. For the guessed test sessions, whether the

IND-CPAF challenge has replaced the original communication, between the tested session and its peer, is unrecognizable from $\mathcal{A}$'s view, by noting that: (1) the distribution of IND-CPAF challenge tuple follows exactly the same distribution of (epk,ect,K) in a real run of the protocol; (2) any event that brings to an association of epk to a different ect causes an abort action by the simulator (see the 4th properties in Lemma 4); (3) the inner state, which includes esk (which the simulator does not know when embeds the CPAF challenge), of the guessed test sessions is never visible to $\mathcal{A}$, since Reveal-state query is not allowed for the tested session or its peer. All actions of embedded simulator, $\mathcal{D}$, after (I_0, s_0) sending its second message (i.e. auth message) do not deviate from the original simulator.

It should be noted that the prerequisite to complete the reduction — by embedding the challenge into the tested session and using the adversary $\mathcal{A}$'s output bit b' as a response to the challenge game — is that the Guess event occurs. Therefore, it is also necessary to discuss whether the difference in the probability of $\hat{\mathcal{S}}$-RAND and $\hat{\mathcal{S}}$-HYBR successfully triggering the Guess event is also negligible. In fact, this can be proven using the exact same procedure of embedding the challenge into the tested session, with the only difference being that instead of observing $\mathcal{A}$'s output bit, one observes whether the Guess event is successfully triggered. If there exists a non-negligible difference in the probability of $\hat{\mathcal{S}}$-RAND and $\hat{\mathcal{S}}$-HYBR triggering the Guess event, one can likewise obtain a non-negligible advantage in the challenge game. Thus we conclude the proof.

□

Lemma 6. *For all Σ_1''-attackers $\mathcal{A}$, $\hat{\mathcal{S}}$-REAL$(\mathcal{A}) \approx \hat{\mathcal{S}}$-RPRF$(\mathcal{A})$, relying on the IND-CPAF security of the underlying KEM.*

Proof. The proof of Lemma 6 and Lemma 5 follow almost identical proof strategies; the only difference is that the K_b obtained from the CPAF challenge of the KEM requires one additional computation of $\mathsf{PRF}_{K_b}(0)$:

4. If (R_0, s_0) receives a response message, R_0 verifies the signature included in it under the identity public key of the initiator (as it appears in the response message). Then R_0 verifies the MAC through recalculation using $k_1 = \mathsf{PRF}_{K_b}(1)$, derived from K_b. If all verifications are valid, $\mathcal{D}$ then makes (R_0, s_0) output a finish message of the form:

   $[s_0, R_0, \mathsf{SIG}_{R_0}(\text{"1"}, s_0, \mathsf{pk}, \mathsf{ct}^*), \mathsf{MAC}_{k_1}(\text{"1"}, s_0, R_0)]$; and R_0 accepts the session key $ssk^* = \mathsf{PRF}_{K_b}(0)$ (as in $\hat{\mathcal{S}}$-REAL), ready to be tested.

5. On incoming finish message to (I_0, s_0), I_0 verifies the signature in it under the identity public key of the responder. Then, the MAC's

verification is conducted in a normal manner for I_0, i.e. recalculating MAC from $k_1 = \mathsf{PRF}_{K_b}(1)$ and comparing it to the received value. If all verifications are valid, I_0 accepts the session key $ssk^* = \mathsf{PRF}_{K_b}(0)$ (as in $\hat{\mathcal{S}}$-REAL), ready to be tested.

□

Lemma 7. *For all Σ_1''-attackers $\mathcal{A}$, $\hat{\mathcal{S}}$-ALLR($\mathcal{A}$) $\approx$ $\hat{\mathcal{S}}$-HYBR($\mathcal{A}$), relying on the PRF security of the underlying PRF.*

Proof. The proof of Lemma 7 requires embedding the access results of the function $f_b(\cdot)$ in the PRF challenge game, replacing the key k_1 used for MAC computation with $f_b(0)$. The complete process of embedding is described as follows, we assume (I_0, s_0) and (R_0, s_0) are the guessed test session and its peer:

1. $\mathcal{D}$ sets $[s_0, \mathsf{pk}]$ as the start message from (I_0, s_0).

2. If (R_0, s_0) receives a response message involving pk then $\mathcal{D}$ produces $[s_0, \mathsf{ct}^*]$ as the response message from (R_0, s_0). Otherwise, if (R_0, s_0) receives a response message with *epk* different than pk then $\mathcal{D}$ conduct the general behavior, i.e. generating *ect* from *epk* and continues.

3. Upon (I_0, s_0) receiving a response message containing *ect*, $\mathcal{D}$ queries $\mathcal{O}_{\mathsf{FC}^*}(ect)$ to obtain the value d. If $d = 0/\perp$, $\mathcal{D}$ aborts; If $d = 1$, then $\mathcal{D}$ generates a response message from (I_0, s_0) in the form $[s_0, I_0, \mathsf{SIG}_{I_0}(\text{"0"}, s_0, \mathsf{ct}^*, \mathsf{pk}), \mathsf{MAC}_{k_1}(\text{"0"}, s_0, I_0)]$, where $k_1 = f_b(1)$.

4. If (R_0, s_0) receives a response message, R_0 verifies the signature included in it under the identity public key of the initiator (as it appears in the response message). Then R_0 verifies the MAC through recalculation using $k_1 = f_b(1)$. If all verifications are valid, $\mathcal{D}$ then makes (R_0, s_0) output a finish message of the form: $[s_0, R_0, \mathsf{SIG}_{R_0}(\text{"1"}, s_0, \mathsf{pk}, \mathsf{ct}^*)$, $\mathsf{MAC}_{k_1}(\text{"1"}, s_0, R_0)]$; and R_0 accepts the random session key ssk^*, ready to be tested.

5. On incoming finish message to (I_0, s_0), I_0 verifies the signature in it under the identity public key of the responder. Then, the MAC's verification is conducted in a normal manner for I_0, i.e. recalculating MAC from $k_1 = f_b(1)$ and comparing it to the received value. If all verifications are valid, I_0 accepts the random session key ssk^*, ready to be tested.

By the definition of the PRF challenge game, when $b = 0$, $f_b(\cdot)$ is a random function, and $f_b(1)$ can be viewed as $k_1 \leftarrow\!\!\$\ \mathcal{K}$, corresponding to $\hat{\mathcal{S}}$-ALLR; when $b = 1$, $f_b(\cdot)$ is a pseudorandom function, and $f_b(1)$ can be viewed as $k_1 \leftarrow \mathsf{PRF}_k(1)$, corresponding to $\hat{\mathcal{S}}$-HYBR (even though $\hat{\mathcal{S}}$-HYBR does not know K).

Again, we need to discuss whether the difference in the probability of $\hat{\mathcal{S}}$-ALLR and $\hat{\mathcal{S}}$-HYBR successfully triggering the GUESS event is negligible. Using the exact same procedure of embedding the challenge into the tested session, and instead of observing $\mathcal{A}$'s output bit, one observes whether the GUESS event is successfully triggered. If there exists a non-negligible difference in the probability of $\hat{\mathcal{S}}$-ALLR and $\hat{\mathcal{S}}$-HYBR triggering the GUESS event, one can likewise obtain a non-negligible advantage in the challenge game. Thus we conclude the proof. □

Lemma 8. *For all Σ_1''-attackers $\mathcal{A}$, $\hat{\mathcal{S}}$-RPRF$(\mathcal{A}) \approx \hat{\mathcal{S}}$-ALLR$(\mathcal{A})$, relying on the PRF security of the underlying PRF.*

Proof. The proof of Lemma 8 follows almost the same strategy as that of Lemma 7, with the only difference being one additional query to $f_b(\cdot)$, replacing ssk^* with $f_b(0)$:

4. If (R_0, s_0) receives a response message, R_0 verifies the signature included in it under the identity public key of the initiator (as it appears in the response message). Then R_0 verifies the MAC through recalculation using $k_1 = f_b(1)$. If all verifications are valid, $\mathcal{D}$ then makes (R_0, s_0) output a finish message of the form: $[s_0, R_0, \mathsf{SIG}_{R_0}(\text{"1"}, s_0, \mathsf{pk}, \mathsf{ct}^*), \mathsf{MAC}_{k_1}(\text{"1"}, s_0, R_0)]$; and R_0 accepts the session key $ssk^* = f_b(0)$, ready to be tested.

5. On incoming finish message to (I_0, s_0), I_0 verifies the signature in it under the identity public key of the responder. Then, the MAC's verification is conducted in a normal manner for I_0, i.e. recalculating MAC from $k_1 = f_b(1)$ and comparing it to the received value. If all verifications are valid, I_0 accepts the session key $ssk^* = f_b(0)$, ready to be tested.

□

Observe that when $\mathcal{A}$ issues a Test query to simulator $\mathcal{S}$: if the hidden bit $b = 0$, then simulator $\hat{\mathcal{S}}$-REAL$(\mathcal{A})$ and simulator $\mathcal{S}$ are equivalent in $\mathcal{A}$'s view; if the hidden bit $b = 1$, then simulator $\hat{\mathcal{S}}$-RAND$(\mathcal{A})$ and simulator $\mathcal{S}$ are equivalent in $\mathcal{A}$'s view. Therefore, one can combine all of the above lemmas to prove the SK-Security of the Σ_1'' protocol.

Lemma 9. *Protocol Σ_1'' satisfies condition P2 of SK-seurity: for all Σ_1''-attackers $\mathcal{A}$,*

$$|\Pr[\mathcal{S}(\mathcal{A}) = 1 \mid b = 0] - \Pr[\mathcal{S}(\mathcal{A}) = 1 \mid b = 1]|$$

is negligible.

Proof. Since $\hat{\mathcal{S}}$ aborts when the GUESS event does not occur, we have

$$\Pr\Big[\hat{\mathcal{S}}(\mathcal{A}) = 1 \;\Big|\; \neg Guess\Big] = \Pr\Big[\hat{\mathcal{S}}(\mathcal{A}) = 0 \;\Big|\; \neg Guess\Big] = 0$$

From this, we can conclude that

$$\begin{aligned}
&\Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A}) = 1\Big] - \Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A}) = 1\Big] \\
&= \Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A}) = 1 \;\Big|\; Guess\Big] \cdot \Pr\Big[Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A})\Big] \\
&+ \Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A}) = 1 \;\Big|\; \neg Guess\Big] \cdot \Pr\Big[\neg Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A})\Big] \\
&- \Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A}) = 1 \;\Big|\; Guess\Big] \cdot \Pr\Big[Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A})\Big] \\
&- \Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A}) = 1 \;\Big|\; \neg Guess\Big] \cdot \Pr\Big[\neg Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A})\Big] \\
&= \Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A}) = 1 \;\Big|\; Guess\Big] \cdot \Pr\Big[Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A})\Big] \\
&- \Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A}) = 1 \;\Big|\; Guess\Big] \cdot \Pr\Big[Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A})\Big] \\
&= \Pr[\mathcal{S}(\mathcal{A}) = 1 \mid b = 0] \cdot \Pr\Big[Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A})\Big] \\
&- \Pr[\mathcal{S}(\mathcal{A}) = 1 \mid b = 1] \cdot \Pr\Big[Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A})\Big]
\end{aligned}$$

Without loss of generality, we may assume that $\Pr[\mathcal{S}(\mathcal{A}) = 1 \mid b = 1] \leq \frac{1}{2}$, and then by rearranging and taking the absolute value of the above expression, we obtain the inequality:

$$\begin{aligned}
&|\Pr[\mathcal{S}(\mathcal{A}) = 1 \mid b = 0] - \Pr[\mathcal{S}(\mathcal{A}) = 1 \mid b = 1]| \\
&\cdot \Pr\Big[Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A})\Big] \\
&\leq \Big|\Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A}) = 1\Big] - \Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A}) = 1\Big]\Big| \\
&+ \Big|\Pr\Big[Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A})\Big] - \Pr\Big[Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A})\Big]\Big| \\
&\cdot \Pr[\mathcal{S}(\mathcal{A}) = 1 \mid b = 1] \\
&\leq \Big|\Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A}) = 1\Big] - \Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A}) = 1\Big]\Big| \\
&+ \frac{1}{2} \cdot \Big|\Pr\Big[Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A})\Big] - \Pr\Big[Guess \text{ in } \hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A})\Big]\Big|
\end{aligned}$$

Furthermore, since the probability of correctly guessing the tested session among at most N_s sessions and N_p parties is at least $\frac{1}{N_s \cdot N_p}$, we have:

$$\begin{aligned}
&|\Pr[\mathcal{S}(\mathcal{A}) = 1 \mid b = 0] - \Pr[\mathcal{S}(\mathcal{A}) = 1 \mid b = 1]| \\
\leq & N_s N_p \cdot (\Big|\Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A}) = 1\Big] - \Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A}) = 1\Big]\Big| \\
& + \frac{1}{2} \cdot \Big|\Pr\Big[\mathit{Guess\ in}\ \hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A})\Big] - \Pr\Big[\mathit{Guess\ in}\ \hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A})\Big]\Big|)
\end{aligned}$$

Combining Lemma 4 through Lemma 8, we obtain $\hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A}) \approx \hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A})$. Consequently, we have that

$$\Big|\Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A}) = 1\Big] - \Pr\Big[\hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A}) = 1\Big]\Big|$$

and

$$\Big|\Pr\Big[\mathit{Guess\ in}\ \hat{\mathcal{S}}\text{-}\mathsf{REAL}(\mathcal{A})\Big] - \Pr\Big[\mathit{Guess\ in}\ \hat{\mathcal{S}}\text{-}\mathsf{RAND}(\mathcal{A})\Big]\Big|$$

are both negligible. Thus we conclude the proof. □

G.2 Proof for Theorem 1

The difference between Σ_1' and Σ_1'' lies in the additional key k_2 introduced in the third and fourth phases for message encryption. This modification does not affect the proof of P1. The proof of P1 and P2's Lemma 4 remain valid in the encrypted version, Σ_1'.

The proof strategy for P2 also remains essentially the same. At the beginning of its run, $\hat{\mathcal{S}}$ additionally chooses a key k_2 used for message encryption. The specification of the ways in which ssk^*, k_1 and k_2 are chosen are correspondingly updated as follows:

$$\begin{array}{lllll}
\hat{\mathcal{S}}\text{-}\mathsf{REAL}: & ssk^* \leftarrow \mathsf{PRF}_{K'}(0), & k_1 \leftarrow \mathsf{PRF}_{K'}(1), & k_2 \leftarrow \mathsf{PRF}_{K'}(2) \\
\hat{\mathcal{S}}\text{-}\mathsf{RPRF}: & ssk^* \leftarrow \mathsf{PRF}_{k}(0), & k_1 \leftarrow \mathsf{PRF}_{k}(1), & k_2 \leftarrow \mathsf{PRF}_{k}(2), k \leftarrow\!\!\$\, \mathcal{K} \\
\hat{\mathcal{S}}\text{-}\mathsf{ALLR}: & ssk^* \leftarrow\!\!\$\, \mathcal{K}, & k_1 \leftarrow\!\!\$\, \mathcal{K} & k_2 \leftarrow\!\!\$\, \mathcal{K} \\
\hat{\mathcal{S}}\text{-}\mathsf{HYBR}: & ssk^* \leftarrow\!\!\$\, \mathcal{K}, & k_1 \leftarrow \mathsf{PRF}_{k}(1), & k_2 \leftarrow \mathsf{PRF}_{k}(2), k \leftarrow\!\!\$\, \mathcal{K} \\
\hat{\mathcal{S}}\text{-}\mathsf{RAND}: & ssk^* \leftarrow\!\!\$\, \mathcal{K}, & k_1 \leftarrow \mathsf{PRF}_{K'}(1), & k_2 \leftarrow \mathsf{PRF}_{K'}(2)
\end{array}$$

where K' represents the encapsulated secret K_e (resp. the decapsulated secret K_d) for the responder (resp. the initiator) side; ssk^* represents the tested session key; k_1 and k_2 are respectively the MAC key and the encryption key in the 3rd or 4th message in Σ_1''.

Below we list the changes in the proof of Σ_1' relative to the proof of Σ_1'':

Lemma 10. *For all Σ_1'-attackers $\mathcal{A}$, $\hat{\mathcal{S}}$-$\mathsf{RAND}(\mathcal{A}) \approx \hat{\mathcal{S}}$-$\mathsf{HYBR}(\mathcal{A})$, relying on the IND-CPAF security of the underlying KEM.*

Proof. The embedding only differs in the following steps, i.e. to additionally derive k_2 and en-/decrypting the last two messages:

3. Upon (I_0, s_0) receiving a response message containing ect, $\mathcal{D}$ queries $\mathcal{O}_{\mathsf{FC}^*}(ect)$ to obtain the value d. If $d = 0/\perp$, $\mathcal{D}$ aborts; If $d = 1$, then $\mathcal{D}$ generates a response message from (I_0, s_0) in the form $[s_0, k_2\{I_0, \mathsf{SIG}_{I_0}(\text{"0"}, s_0, \mathsf{ct}^*, \mathsf{pk}), \mathsf{MAC}_{k_1}(\text{"0"}, s_0, I_0)\}]$, with $k_1 = \mathsf{PRF}_{K_b}(1)$, $k_2 = \mathsf{PRF}_{K_b}(2)$ and K_b being the third element in the challenge.

4. If (R_0, s_0) receives a response message, R_0 first decrypts the ciphertext using $k_2 = \mathsf{PRF}_{K_b}(2)$. If AEAD verification passes and outputs a valid plaintext, R_0 verifies the signature in it under the identity public key of the initiator (as it appears in the response message). Then R_0 verifies the MAC through recalculation using $k_1 = \mathsf{PRF}_{K_b}(1)$, derived from K_b. If all verifications are valid, (R_0, s_0) outputs a finish message of the form: $[s_0, k_2\{R_0, \mathsf{SIG}_{R_0}(\text{"1"}, s_0, \mathsf{pk}, \mathsf{ct}^*), \mathsf{MAC}_{k_1}(\text{"1"}, s_0, R_0)\}]$, where $k_2 = \mathsf{PRF}_{K_b}(2)$; and R_0 accepts the random session key ssk^* (as in $\hat{\mathcal{S}}$-RAND), ready to be tested.

5. On incoming finish message to (I_0, s_0), I_0 decrypts the ciphertext using $k_2 = \mathsf{PRF}_{K_b}(2)$. If AEAD verification passes and outputs a valid plaintext, then I_0 verifies the signature in it under the identity public key of the responder. Then, the MAC's verification is conducted in a normal manner for I_0, i.e. recalculating MAC from $k_1 = \mathsf{PRF}_{K_b}(1)$ and comparing it to the received value. If all verifications are valid, I_0 accepts the random session key ssk^* (as in $\hat{\mathcal{S}}$-RAND), ready to be tested.

□

Lemma 11. *For all Σ_1'-attackers $\mathcal{A}$, $\hat{\mathcal{S}}$-$\mathsf{REAL}(\mathcal{A}) \approx \hat{\mathcal{S}}$-$\mathsf{RPRF}(\mathcal{A})$, relying on the IND-CPAF security of the underlying KEM.*

Proof. The proof of Lemma 11 and Lemma 10 follow almost identical proof strategies; the only difference is that the K_b obtained from the CPAF challenge of the KEM requires one additional computation of $\mathsf{PRF}_{K_b}(0)$:

4. If (R_0, s_0) receives a response message, R_0 first decrypts the ciphertext using $k_2 = \mathsf{PRF}_{K_b}(2)$. If AEAD verification passes and outputs a valid

plaintext, R_0 verifies the signature in it under the identity public key of the initiator (as it appears in the response message). Then R_0 verifies the MAC through recalculation using $k_1 = \mathsf{PRF}_{K_b}(1)$, derived from K_b. If all verifications are valid, (R_0, s_0) outputs a finish message of the form: $[s_0, k_2\{R_0, \mathsf{SIG}_{R_0}(\text{"1"}, s_0, \mathsf{pk}, \mathsf{ct}^*), \mathsf{MAC}_{k_1}(\text{"1"}, s_0, R_0)\}]$, where $k_2 = \mathsf{PRF}_{K_b}(2)$; and R_0 accepts the session key $ssk^* = \mathsf{PRF}_{K_b}(0)$ (as in $\hat{\mathcal{S}}$-REAL), ready to be tested.

5. On incoming finish message to (I_0, s_0), I_0 decrypts the ciphertext using $k_2 = \mathsf{PRF}_{K_b}(2)$. If AEAD verification passes and outputs a valid plaintext, then I_0 verifies the signature in it under the identity public key of the responder. Then, the MAC's verification is conducted in a normal manner for I_0, i.e. recalculating MAC from $k_1 = \mathsf{PRF}_{K_b}(1)$ and comparing it to the received value. If all verifications are valid, I_0 accepts the session key $ssk^* = \mathsf{PRF}_{K_b}(0)$ (as in $\hat{\mathcal{S}}$-REAL), ready to be tested.

□

Lemma 12. *For all Σ_1'-attackers $\mathcal{A}$, $\hat{\mathcal{S}}$-$\mathsf{ALLR}(\mathcal{A}) \approx \hat{\mathcal{S}}$-$\mathsf{HYBR}(\mathcal{A})$, relying on the PRF security of the underlying PRF.*

Proof. The proof of Lemma 12 follows almost the same strategy as that of Lemma 7, with the only difference being one additional query to $f_b(\cdot)$ to calculate $k_2 = f_b(2)$:

3. Upon (I_0, s_0) receiving a response message containing ect, $\mathcal{D}$ queries $\mathcal{O}_{\mathsf{FC}^*}(ect)$ to obtain the value d. If $d = 0/\perp$, $\mathcal{D}$ aborts; If $d = 1$, then $\mathcal{D}$ generates a response message from (I_0, s_0) in the form $[s_0, k_2\{I_0, \mathsf{SIG}_{I_0}(\text{"0"}, s_0, \mathsf{ct}^*, \mathsf{pk}), \mathsf{MAC}_{k_1}(\text{"0"}, s_0, I_0)\}]$, with $k_1 = f_b(1)$, $k_2 = f_b(2)$ and K_b being the third element in the challenge.

4. If (R_0, s_0) receives a response message, R_0 first decrypts the ciphertext using $k_2 = f_b(2)$. If AEAD verification passes and outputs a valid plaintext, R_0 verifies the signature in it under the identity public key of the initiator (as it appears in the response message).

   Then R_0 verifies the MAC through recalculation using $k_1 = f_b(1)$, derived from K_b. If all verifications are valid, (R_0, s_0) outputs a finish message of the form:

   $[s_0, k_2\{R_0, \mathsf{SIG}_{R_0}(\text{"1"}, s_0, \mathsf{pk}, \mathsf{ct}^*), \mathsf{MAC}_{k_1}(\text{"1"}, s_0, R_0)\}]$, where $k_2 = f_b(2)$; and R_0 accepts the random session key ssk^*, ready to be tested.

5. On incoming finish message to (I_0, s_0), I_0 decrypts the ciphertext using $k_2 = f_b(2)$. If AEAD verification passes and outputs a valid plaintext, then I_0 verifies the signature in it under the identity public key of the responder. Then, the MAC's verification is conducted in a normal manner for I_0, i.e. recalculating MAC from $k_1 = f_b(1)$ and comparing it to the received value. If all verifications are valid, I_0 accepts the random session key ssk^*, ready to be tested.

□

Lemma 13. *For all Σ_1'-attackers $\mathcal{A}$, $\hat{\mathcal{S}}$-$\mathsf{RPRF}(\mathcal{A}) \approx \hat{\mathcal{S}}$-$\mathsf{ALLR}(\mathcal{A})$, relying on the PRF security of the underlying PRF.*

Proof. The proof of Lemma 13 follows almost the same strategy as that of Lemma 12, with the only difference being one additional query to $f_b(\cdot)$, replacing ssk^* with $f_b(0)$:

4. If (R_0, s_0) receives a response message, R_0 first decrypts the ciphertext using $k_2 = f_b(2)$. If AEAD verification passes and outputs a valid plaintext, R_0 verifies the signature in it under the identity public key of the initiator (as it appears in the response message).

   Then R_0 verifies the MAC through recalculation using $k_1 = f_b(1)$, derived from K_b. If all verifications are valid, (R_0, s_0) outputs a finish message of the form:

   $[s_0, k_2\{R_0, \mathsf{SIG}_{R_0}(\text{"1"}, s_0, \mathsf{pk}, \mathsf{ct}^*), \mathsf{MAC}_{k_1}(\text{"1"}, s_0, R_0)\}]$, where $k_2 = f_b(2)$; and R_0 accepts the session key $ssk^* = f_b(0)$, ready to be tested.

5. On incoming finish message to (I_0, s_0), I_0 decrypts the ciphertext using $k_2 = f_b(2)$. If AEAD verification passes and outputs a valid plaintext, then I_0 verifies the signature in it under the identity public key of the responder. Then, the MAC's verification is conducted in a normal manner for I_0, i.e. recalculating MAC from $k_1 = f_b(1)$ and comparing it to the received value. If all verifications are valid, I_0 accepts the session key $ssk^* = f_b(0)$, ready to be tested.

□

Lemma 14. *Protocol Σ_1' satisfies condition P2 of SK-seurity: for all Σ_1'-attackers $\mathcal{A}$,*

$$|\Pr[\mathcal{S}(\mathcal{A}) = 1 \mid b = 0] - \Pr[\mathcal{S}(\mathcal{A}) = 1 \mid b = 1]|$$

is negligible.

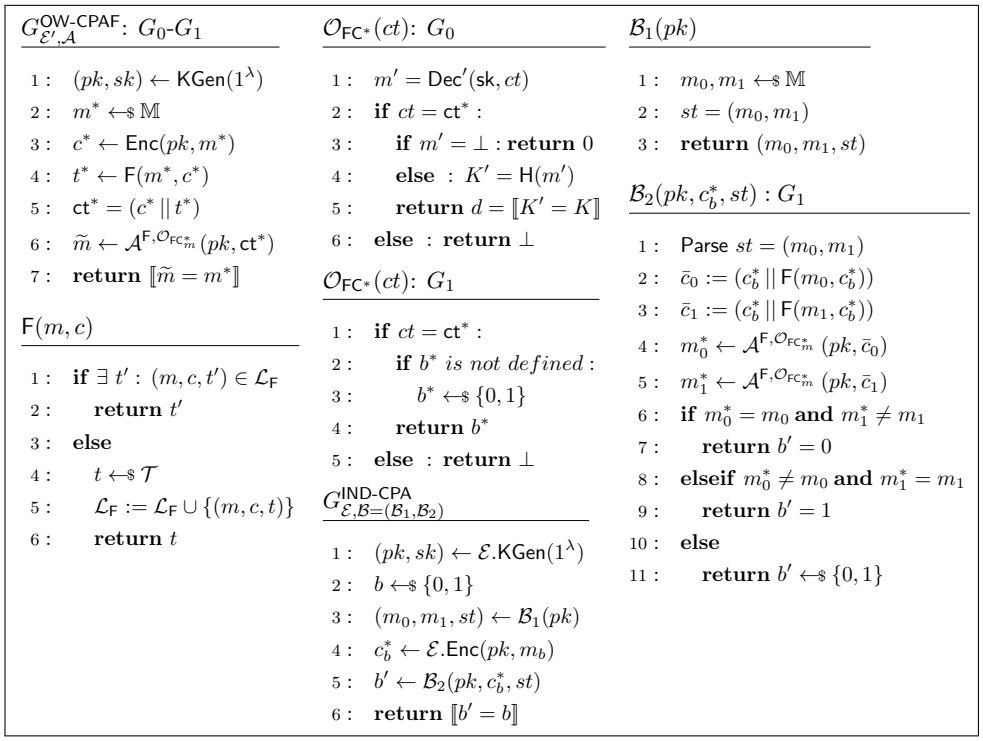


Figure 7: OW-CPAF PKE to IND-CPA PKE reduction

Proof. Combining Lemma 10 through Lemma 13, and following exactly the same proof strategy as in Lemma 9, we can conclude the proof. □

H Omitted Proofs for the KEM Construction

H.1 Proof for Lemma 2

Proof. For the correctness of $\mathcal{E}'$, simply note that this transformation does not change the behavior of correct decryption; while for those decryption returned with $m' \neq m$, Dec' "absorbs" most of them into abort case by checking whether $\mathsf{F}(m', c) = \mathsf{F}(m, c) = t$. Therefore, those ciphertexts that originally cause decryption failure will still fail, and the converse also still holds.

In Figure 7, Game G_0 is the real game for the OW-CPAF adversary $\mathcal{A}$ against PKE $\mathcal{E}'$. Now, the reduction game hops are described below.

GAME G_1. In game G_1, the only difference from G_0 is that the dependence of $\mathcal{O}_{\mathsf{FC}^*_m}$ on the private key sk is removed; instead, a random coin b^* is tossed to guess the correct output of $\mathcal{O}_{\mathsf{FC}^*_m}(ct^*)$, which succeeds with probability $1/2$. When b^* is guessed correctly, G_1 is identical to G_0 from the adversary's perspective, and therefore the probability that the adversary wins in G_1 is at least half of that in G_0: $\Pr\big[G_0^{\mathcal{A}} = 1\big] \leq 2 \cdot \Pr\big[G_1^{\mathcal{A}} = 1\big]$.

To bound the probability that the adversary wins in G_1, we construct an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the $\mathsf{IND\text{-}CPA}$ security of $\mathcal{E}$, as illustrated in Figure 7. The strategy of $\mathcal{B}_1$ for selecting the challenge plaintext pair is to randomly sample two mutually independent messages $m_0, m_1 \leftarrow\!\!\$\, \mathbb{M}$. After returning this pair to the challenger, $\mathcal{B}$ receives the challenge ciphertext $c_b^* = \mathcal{E}.\mathsf{Enc}(pk, m_b)$. Using m_0 and m_1 combined respectively with c_b^* as the input to the random oracle F, $\mathcal{B}$ constructs two ciphertexts in the format of $\mathcal{E}'$:

$$\bar{c}_0 = c_b^* \,\|\, \mathsf{F}(m_0, c_b^*), \qquad \bar{c}_1 = c_b^* \,\|\, \mathsf{F}(m_1, c_b^*).$$

Together with the challenge public key pk, these are used as inputs to invoke the adversary $\mathcal{A}$ twice, with $\mathcal{B}$ simulating the F and $\mathcal{O}_{\mathsf{FC}^*_m}$ oracles for $\mathcal{A}$ in each invocation:

- When $b = 0$: in the first invocation, $\mathcal{A}^{\mathsf{F},\mathcal{O}_{\mathsf{FC}^*_m}}\big(pk,\, \bar{c}_0 = [c_0^* \,\|\, \mathsf{F}(m_0, c_0^*)]\big)$ is equivalent to being challenged in G_1; in the second invocation, since m_0 and m_1 are mutually independent, if $\mathcal{A}^{\mathsf{F},\mathcal{O}_{\mathsf{FC}^*_m}}\big(pk,\, \bar{c}_1 = [c_0^* \,\|\, \mathsf{F}(m_1, c_0^*)]\big)$ outputs m_1, this amounts to finding a preimage of a random oracle output (since the adversary can freely sample any m_0' and encrypt it under $\mathcal{E}$ to simulate $\mathcal{A}^{\mathsf{F},\mathcal{O}_{\mathsf{FC}^*_m}}\big(pk,\, \bar{c}_1 = [c_0^* \,\|\, \mathsf{F}(m_1, c_0^*)]\big)$); by the properties of the random oracle, we have

$$\Pr\Big[\mathcal{A}^{\mathsf{F},\mathcal{O}_{\mathsf{FC}^*_m}}\big(pk,\, \bar{c}_1 = [c_0^* \,\|\, \mathsf{F}(m_1, c_0^*)]\big) = m_1\Big] \leq \frac{q_{\mathsf{F}}}{|\mathcal{T}|}.$$

- When $b = 1$: the situation is reversed. In the first invocation,

$$\Pr\Big[\mathcal{A}^{\mathsf{F},\mathcal{O}_{\mathsf{FC}^*_m}}\big(pk,\, \bar{c}_0 = [c_0^* \,\|\, \mathsf{F}(m_0, c_0^*)]\big) = m_0\Big] \leq \frac{q_{\mathsf{F}}}{|\mathcal{T}|};$$

  in the second invocation, $\mathcal{A}\big(pk,\, \bar{c}_1\big)$ is equivalent to being challenged in G_1.

Adversary $\mathcal{B}$ then verifies the returned values of the two invocations of $\mathcal{A}$:

- If the returned value of the first invocation satisfies $m_0^* = m_0$ and the returned value of the second invocation satisfies $m_1^* \neq m_1$, then $\mathcal{B}$ outputs bit $b' = 0$ in the $G^{\mathsf{IND\text{-}CPA}}$ challenge;
- If the returned value of the first invocation satisfies $m_0^* \neq m_0$ and the returned value of the second invocation satisfies $m_1^* = m_1$, then $\mathcal{B}$ outputs bit $b' = 1$ in the $G^{\mathsf{IND\text{-}CPA}}$ challenge;
- If neither of the returned values is equal to m_0 or m_1, or if both returned values respectively matches with m_0 and m_1, then $\mathcal{B}$ randomly samples an output bit $b' \leftarrow\!\!\$\ \{0,1\}$ in the $G^{\mathsf{IND\text{-}CPA}}$ challenge.

Regardless of the challenge bit b, $\mathcal{A}$ is effectively challenged once in G_1 and once tasked with finding a preimage of a random oracle output. And either of the following 2 cases will happen: (1) $\mathcal{A}$ wins only in the former challenge, G_1, and $\mathcal{B}$ always output the correct challenge bit; or (2) $\mathcal{B}$ guesses b correctly with probability $\frac{1}{2}$. Therefore, we have:

$$
\begin{aligned}
&\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{E},\mathcal{B}} = \left|\Pr[b' = b] - \frac{1}{2}\right| \\
&= \left|\Pr[G_1 = 1] \cdot \left(1 - \Pr\left[\mathcal{A}^{\mathsf{F},\mathcal{O}_{\mathsf{FC}^*_m}}\left(pk,\, \bar{c}_{1\oplus b}\right) = m_{1\oplus b}\right]\right)\right. \\
&\left. + \frac{1}{2}\left(1 - \Pr[G_1 = 1] \cdot \left(1 - \Pr\left[\mathcal{A}^{\mathsf{F},\mathcal{O}_{\mathsf{FC}^*_m}}\left(pk,\, \bar{c}_{1\oplus b} = m_{1\oplus b}\right]\right]\right)\right) - \frac{1}{2}\right| \\
&= \frac{1}{2} \cdot \Pr[G_1 = 1] \cdot \left(1 - \Pr\left[\mathcal{A}^{\mathsf{F},\mathcal{O}_{\mathsf{FC}^*_m}}\left(pk,\, \bar{c}_{1\oplus b} = m_{1\oplus b}\right]\right]\right) \\
&\geq \frac{1}{2} \cdot \Pr[G_1 = 1] \cdot \left(1 - \frac{q_F}{|\mathcal{T}|}\right)
\end{aligned}
$$

Rearranging, we obtain:

$$\Pr[G_1 = 1] \leq 2\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{,\mathcal{E},\mathcal{B}} / \left(1 - \frac{q_F}{|\mathcal{T}|}\right)$$

and: $\mathsf{Adv}^{\mathsf{OW\text{-}CPAF}}_{\mathcal{E}',\mathcal{A}} =$

$$= \Pr[G_0 = 1] \leq 2\Pr[G_1 = 1] \leq \left(1 - \frac{q_F}{|\mathcal{T}|}\right)^{-1} \cdot 4\ \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{E},\mathcal{B}}$$

□

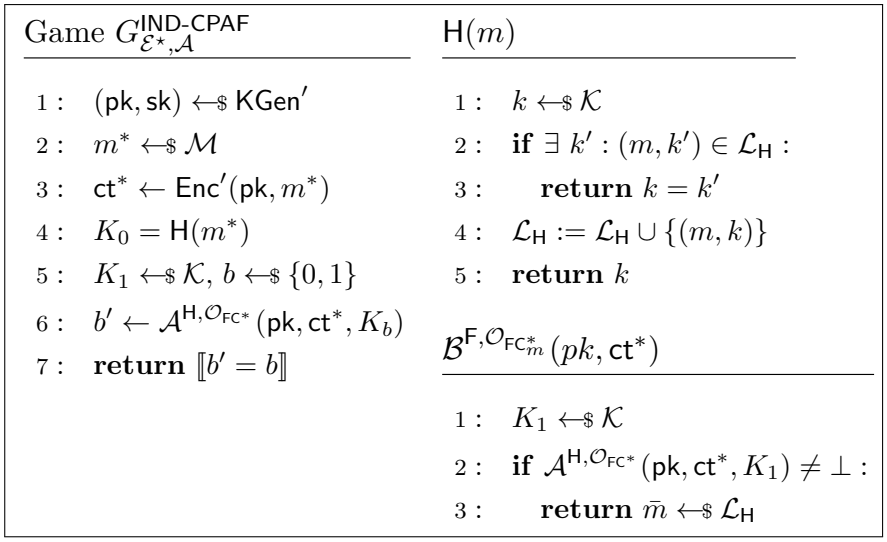


Figure 8: IND-CPAF KEM to OW-CPAF PKE reduction.

H.2 Proof for Lemma 3

Proof. As illustrated in Figure 8, suppose an adversary $\mathcal{A}$ wins the IND-CPAF game with non-negligible advantage $\mathsf{Adv}^{\mathsf{IND\text{-}CPAF}}_{\mathcal{E}^\star} = \left|\Pr\left[G^{\mathsf{IND\text{-}CPAF}}_{\mathcal{E}^\star,\mathcal{A}} = b\right] - \frac{1}{2}\right|$, then we invoke $\mathcal{A}$ to construct an adversary $\mathcal{B}$ attacking the OW-CPAF game in Figure 7, and adversary $\mathcal{B}$ simulates the game $G^{\mathsf{IND\text{-}CPAF}}_{\mathcal{E}^\star,\mathcal{A}}$ with $b = 1$ for adversary $\mathcal{A}$. Define the event Flag as the event that $\mathcal{A}$ queries the random oracle H with m^*. By the properties of random oracle, conditioned on Flag not being triggered, $K_0 = \mathsf{H}(m^*)$ and $K_1 \leftarrow\!\$\ \mathcal{K}$ are indistinguishable to $\mathcal{A}$, and hence the probability that $\mathcal{A}$ correctly guesses b is the same regardless of whether $b = 0$ or $b = 1$:

$$\begin{aligned}
&\Pr\left[\mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathsf{FC}^*}}(\mathsf{pk}, \mathsf{ct}^*, K_1) = b \;\middle|\; \neg\mathsf{Flag} \ \wedge\ b = 1\right] \\
=&\Pr\left[\mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathsf{FC}^*}}(\mathsf{pk}, \mathsf{ct}^*, K_0) = b \;\middle|\; \neg\mathsf{Flag} \ \wedge\ b = 0\right] \\
=&\Pr\left[\mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathsf{FC}^*}}(\mathsf{pk}, \mathsf{ct}^*, K_b) = b \;\middle|\; \neg\mathsf{Flag}\right] = \frac{1}{2}
\end{aligned}$$

Furthermore, we have:

$$\begin{aligned}
&\mathsf{Adv}^{\mathsf{IND\text{-}CPAF}}_{\mathcal{E}^\star,\mathcal{A}} = \left|\Pr[b' = b] - \frac{1}{2}\right| \\
&= \left|\Pr[\mathsf{Flag}] \cdot \Pr\left[\mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathsf{FC}^*}}(\mathsf{pk}, \mathsf{ct}^*, K_b) = b \;\middle|\; \mathsf{Flag}\right]\right. \\
&\left. + \Pr[\neg\mathsf{Flag}] \cdot \Pr\left[\mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathsf{FC}^*}}(\mathsf{pk}, \mathsf{ct}^*, K_b) = b \;\middle|\; \neg\mathsf{Flag}\right] - \frac{1}{2}\right| \\
&= \left|\Pr[\mathsf{Flag}] \cdot \Pr\left[\mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathsf{FC}^*}}(\mathsf{pk}, \mathsf{ct}^*, K_b) = b \;\middle|\; \mathsf{Flag}\right]\right. \\
&\left. + (1 - \Pr[\mathsf{Flag}]) \cdot \frac{1}{2} - \frac{1}{2}\right| \\
&= \left|\Pr\left[\mathcal{A}^{\mathsf{H},\mathcal{O}_{\mathsf{FC}^*}}(\mathsf{pk}, \mathsf{ct}^*, K_b) = b \;\middle|\; \mathsf{Flag}\right] - \frac{1}{2}\right| \cdot \Pr[\mathsf{Flag}] \\
&\leq \Pr[\mathsf{Flag}]
\end{aligned}$$

When Flag is triggered, it means that $\mathcal{A}$ has queried H with m^* at least once. Hence, by uniformly sampling m' from the query list $\mathcal{L}_{\mathsf{H}}$, the probability that $m' = m^*$ is at least $\frac{1}{q_{\mathsf{H}}}$, yielding:

$$\Pr[\mathsf{Flag}] \cdot \frac{1}{q_{\mathsf{H}}} \leq \mathsf{Adv}^{\mathsf{OW\text{-}CPAF}}_{\mathcal{E}',\mathcal{B}}$$

and:

$$\mathsf{Adv}^{\mathsf{IND\text{-}CPAF}}_{\mathcal{E}^\star,\mathcal{A}} \leq \Pr[\mathsf{Flag}] \leq q_{\mathsf{H}} \cdot \mathsf{Adv}^{\mathsf{OW\text{-}CPAF}}_{\mathcal{E}',\mathcal{B}}$$

□