# Under What Conditions Is Encrypted Key Exchange Actually Secure?

Jake Januzelli[1], Lawrence Roy[2], and Jiayu Xu[3]

[1]Oregon State University, januzelj@oregonstate.edu
[2]Aarhus University, ldr709@gmail.com
[3]Oregon State University, xujiay@oregonstate.edu

**Abstract**

A Password-Authenticated Key Exchange (PAKE) protocol allows two parties to agree upon a cryptographic key, in the setting where the only secret shared in advance is a low-entropy password. The standard security notion for PAKE is in the Universal Composability (UC) framework. In recent years there have been a large number of works analyzing the UC-security of Encrypted Key Exchange (EKE), the very first PAKE protocol, and its One-encryption variant (OEKE), both of which compile an unauthenticated Key Agreement (KA) protocol into a PAKE.

In this work, we present a comprehensive and thorough study of the UC-security of both EKE and OEKE in the *most general* setting and using the *most efficient* building blocks:

1. We show that among the seven existing results on the UC-security of (O)EKE, six are flawed;
2. We show that for (O)EKE to be UC-secure, the underlying KA protocol needs to satisfy the properties of *strong pseudorandomness*, *pseudorandom non-malleability*, and *collision resistance*, all of which are missing in existing works;
3. We give UC-security proofs for EKE and OEKE using Programmable-Once Random Function (POPF), which is the most efficient instantiation to date and is around 4 times faster than the standard instantiation using Ideal Cipher (IC).

Our results in particular allow for PAKE constructions from post-quantum KA protocols such as Kyber. We also give a security analysis of POPF in a new composition framework called *almost UC*, which we believe is interesting in its own right.

## Contents

# 1 Introduction

A *Password-Authenticated Key Exchange (PAKE)* protocol allows two parties to agree upon a cryptographic key in the setting where the only information shared in advance is a low-entropy password. Crucially, such protocols must be secure against man-in-the-middle adversaries that can arbitrarily modify the protocol messages sent between the two parties. PAKE protocols — and their extensions such as asymmetric PAKE and threshold PAKE — provide significant advantages over the traditional "password-over-TLS" approach to authentication, in that PAKE does not require transmission of public keys and thus does not need a PKI. Recent years have witnessed an increasing amount of interest in PAKE from both academia and industry, and with the standardization process by the IETF in 2019–20 [Cry20], PAKE protocols are gradually replacing password-over-TLS in practice.

**Security notions for PAKE.** Since passwords have low entropy, an inevitable attack that must be taken into account by any security definition for PAKE is *online guessing*, where the adversary guesses a password $\mathsf{pw}^*$ and executes Alice's algorithm on $\mathsf{pw}^*$ while communicating with Bob; if $\mathsf{pw}^*$ is indeed Bob's password, then the adversary learns Bob's session key. The security requirement of PAKE essentially says that online guessing is the only possible venue of attack; if we assume the password is uniformly distributed over the dictionary $\mathsf{Dict}$, then the adversary's advantage should only be negligibly greater than $q/|\mathsf{Dict}|$ for $q$ online sessions. This is the basis of the game-based security notion for PAKE [BPR00]. While this notion achieves a basic level of PAKE security, it does not provide any security guarantee under composition; in particular, it fails to model password reuse across multiple accounts which is (unfortunately) all too common in real life.

For this reason, the game-based PAKE security definition has been superseded by the definition in the Universal Composability (UC) framework [CHK+05], where security remains under arbitrary composition. Password reuse is addressed by letting the environment choose the password, rather than assuming a specific distribution over the dictionary. Over the years the UC notion has become the de facto security standard for PAKE; for example, all candidates in the second round of the IETF standardization competition have a UC security analysis [AHH21, ABB+20, JKX18, HL19].

**Encrypted Key Exchange (EKE).** The very first PAKE protocol ever proposed is *Encrypted Key Exchange (EKE)* by Bellovin and Merritt in 1992 [BM92]. It compiles any unauthenticated Key Agreement (KA) protocol into a PAKE by encrypting all messages using a private-key encryption scheme with the password as the key (and the receiver can decrypt and recover the message in the underlying KA protocol if it holds the correct password). In the standard instantiation of EKE, the encryption scheme is Ideal Cipher (IC). When instantiated with a 2-round KA protocol, we immediately obtain a PAKE protocol that also has 2 rounds.[1]

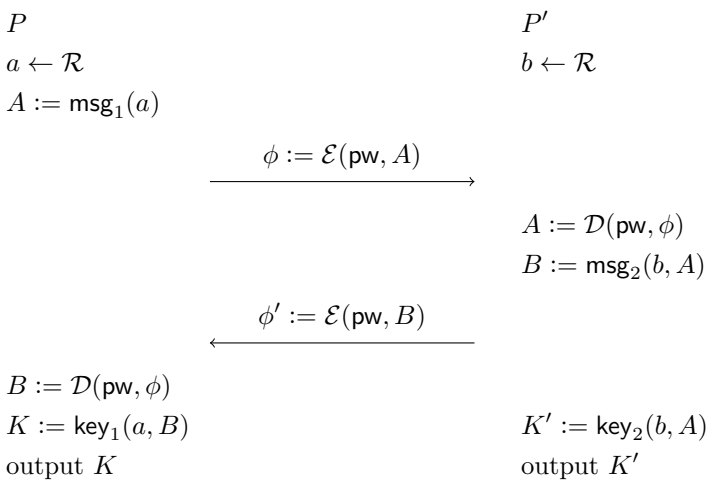


Figure 1: EKE with a 2-round KA protocol

Aside of its historical significance, EKE continues to play an important role in PAKE design. Protocols such as SPAKE1 and SPAKE2 [AP05] are essentially EKE instantiated with specific private-key encryption schemes and KA protocols, and others such as CPace [HL19] inherit some

[1]The term "round" has various definitions in the literature. Here by 2-round we mean 2-flow, i.e., $P$ sends a message to $P'$, and $P'$, upon receiving the message from $P$, sends a message back. If the two messages can be sent simultaneously, we call the protocol 1-simultaneous round.

design ideas from EKE. It is fair to say that EKE and its variants form one of the two major paradigms for PAKE protocols (the other is Smooth Projective Hash Function (SPHF)-based PAKEs [GL03, GK10] which are generally less efficient).

**One-encryption EKE (OEKE).** The following variant of EKE first appeared in [BCP03]: only the $P$-to-$P'$ message is encrypted, and upon receiving the message from $P$, $P'$ sends a *plaintext* KA message to $P$ together with an authenticator that is the second part of $K'$ (where $K'$ is KA key of $P'$); the session key of $P'$ is the first part of $K'$.[2] $P$ computes its KA key $K$ and checks if the authenticator is the second part of $K$, and if so, outputs the first part of $K$ as its session key. This protocol is called OEKE (O for "One-encryption").

OEKE requires that the underlying KA protocol have key length longer than PAKE session key length $\kappa$. If we only have a KA protocol whose key $K$ is $\kappa$-bit long, one way to implement OEKE is to define the session key as $\mathsf{PRF}_K(0)$ and the authenticator as $\mathsf{PRF}_K(1)$ (where $\mathsf{PRF}$ is a PRF with $\kappa$-bit outputs) — that is, we compile the KA protocol into another KA protocol whose key is $\mathsf{PRF}_K(0) \,\|\, \mathsf{PRF}_K(1)$, and use the latter in OEKE. We call this protocol OEKE-PRF, and if the PRF is defined as $\mathsf{PRF}_K(x) = H(K, x)$ (where $H$ is an RO), we call it OEKE-RO. All existing works on the UC-security of OEKE follow the OEKE-PRF paradigm [SGJ23, BCP+23].[3]

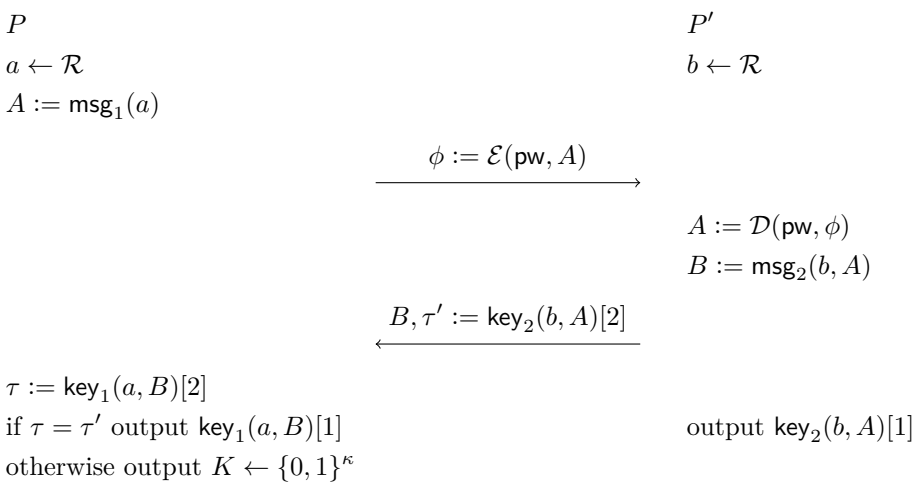


Figure 2: OEKE with a 2-round KA protocol. This version requires a KA protocol with long key. In OEKE-PRF the KA key $K$ has length $\kappa$, PAKE session key is $\mathsf{PRF}_K(0)$, and $\tau = \mathsf{PRF}_K(1)$. In OEKE-RO $\mathsf{PRF}_K(x) = H(K, x)$

While OEKE might not be as round-efficient as EKE (if the underlying KA protocol is 1-simultaneous round such as Diffie-Hellman), its advantage is that it avoids one instance of IC and is thus more efficient in computation cost than EKE. Furthermore, OEKE achieves *one-sided explicit authentication*, where $P$ can tell if it arrives at the same session key as $P'$, and may abort if it does not. This property is not satisfied by EKE or the standard UC PAKE functionality.

[2] Since each party has an output key in the KA protocol and an output key in the PAKE protocol (the latter being its eventual output), it might be confusing what a "key" refers to. Throughout this work we use the terms "KA key" and "(PAKE) session key" to distinguish them.

[3] Formally, [SGJ23] and [BCP+23] use a Key Encapsulation Mechanism (KEM) rather than a KA protocol as a building block, but a KEM is equivalent to a 2-round KA protocol: the first KA message is the KEM public key, the second KA message is the KEM ciphertext, and the KA key is the KEM key. (See, e.g., [SGJ23, Section 2.2].) [SGJ23] calls OEKE "EKE-KEM".

**Remark 1.1.** *One might (rightfully) ask why we consider OEKE-PRF at all, since it is natural to assume the PRF is an RO given that we already rely on idealized models such as IC. Interestingly, the UC-security of OEKE-PRF — when instantiated with Diffie-Hellman KA — requires more properties of the underlying KA than OEKE-RO; the security analysis in [SGJ23] conflates OEKE-PRF and OEKE-RO and causes significant confusion to its readers, so we believe it is helpful to separate these two variants while discussing existing works (while our own analysis considers OEKE, which covers both OEKE-PRF and OEKE-RO). See Sect. 1.1 for more details.*

## 1.1 Existing Security Analyses of (O)EKE

Despite its deceiving simplicity and its pivotal role in over 30 years of study of PAKE, a satisfactory formal security analysis of EKE has long been elusive. The original EKE paper in 1992 [BM92] does not provide a security proof (in fact, even a formal security definition did not exist until 2000 [BPR00]). [BPR00], in addition to its main contribution of proposing the game-based security definition for PAKE, proves that EKE satisfies this definition under the Computational Diffie-Hellman (CDH) assumption if the underlying KA protocol is hashed Diffie-Hellman and the underlying symmetric encryption scheme is IC. However, this result suffers from three drawbacks:

- The result assumes a specific KA protocol (hashed Diffie-Hellman) is used in EKE, and it is not immediately clear how to generate the result to arbitrary KA protocols — that is, it is unclear what properties the KA protocol needs to satisfy for EKE to be secure. We will see that in the general case, it is surprisingly difficult to get the exact necessary properties of the KA protocol right. Pinpointing such properties is especially beneficial if we want to use a post-quantum KA protocol to achieve a PAKE under post-quantum assumptions — which was not a main concern when [BPR00] was published but has gained much importance and attention since then.
- The result assumes that the underlying symmetric encryption scheme is IC. While it is known that a 8-round Feistel network is indifferentiable from an IC in the Random Oracle Model (ROM) [DS16], using an IC in EKE results in a PAKE protocol with significant cost. In particular, since the KA protocol is supposed to be hashed Diffie-Hellman, an IC *onto a group* is needed, which in turn requires 4 RO-hash onto a group operations which are inefficient.
- Finally, the result is in the game-based setting, which as we have mentioned has some critical drawbacks and has been superseded by the UC definition.

[BCP03] provides a security proof for OEKE, which suffers from exactly the three deficiencies above: the underlying KA protocol must be hashed Diffie-Hellman, the underlying symmetric encryption scheme must be IC, and it is in the game-based setting.

| | proposed | game-based analysis | UC analysis |
|---|---|---|---|
| EKE | 1992 [BM92] | 2000 [BPR00] | 2018 [DHP+18] |
| OEKE | 2003 [BCP03] | 2003 [BCP03] | 2023 [SGJ23, BCP+23] |

Table 1: Timeline of security analyses of (any instantiation of) EKE and OEKE

**UC analyses of (O)EKE.** It was not until 18 years later that EKE was formally proven secure in the UC framework [DHP+18]. (For OEKE, we had to wait 20 years [SGJ23, BCP+23].) Since

then, there have been a number of UC analyses of (O)EKE, each with its own instantiation. Below we give a brief summary of existing works in this domain:

- [DHP+18, Theorem 6] shows that EKE is UC-secure under CDH if the underlying KA protocol is hashed Diffie-Hellman (except that the PAKE transcript is also included in the RO hash) and the underlying encryption scheme is IC. This result can be seen as rendering [BPR00] in the UC setting and inherits the first two weaknesses of [BPR00], namely it uses a specific (and pre-quantum) KA protocol and an IC. While the security statement is correct, we will see that its proof contains a flawed reduction in the hybrids, resulting in an incorrect statement of the tightness of the security.

- [MRR20, Theorem 10] claims that EKE is UC-secure provided that the underlying KA protocol is secure and *pseudorandom* (the KA messages are indistinguishable from random) and the underlying encryption scheme is *Programmable-Once Public Function (POPF)*, which can be seen as a generalization of IC. [MRR20] introduces the necessary (game-based) properties of the POPF for EKE to be UC-secure. Using a POPF instead of an IC drastically reduces costs, as a POPF can be instantiated by a 2-round Feistel network (as opposed to 8-round in IC). Since [MRR20] uses *any* secure and pseudorandom KA protocol, it allows for an instantiation of EKE under post-quantum assumptions. Unfortunately, as we will see below, the main result in [MRR20] is incorrect in three different ways.

- [SGJ23, Theorem 2] also claims UC-security of EKE assuming the underlying KA protocol is secure and pseudorandom, and the underlying encryption scheme is *Half Ideal Cipher (HIC)*. [SGJ23] proposes a UC definition for HIC and shows that it is realized by a 2-round Feistel network, except that the XOR operation in the second round is replaced by an IC *over bistrings*. Thus, while HIC is not as efficient as the POPF in [MRR20], it achieves UC-security and is easier to use in other contexts. However, we will show that this result is incorrect in two different ways.
  In addition to the result above, [SGJ23, Theorem 3] claims that a certain variant of OEKE is UC-secure, again using HIC and a secure and pseudorandom KA protocol. Unfortunately, it is not entirely clear whether the protocol analyzed in [SGJ23] is OEKE-PRF or OEKE-RO, and there seems to be a mismatch between the theorem statement and its proof, as mentioned in [SGJ23, footnote 14]:

  > *The proof below assumes a version of the protocol which uses* $\mathsf{prf}(K, \cdot)$ *to derive the authenticator* $\tau$ *and the session key [...] This version of the protocol requires an additional assumption on KEM. We will update the proof shortly to reflect the modified protocol and get rid of the additional assumption.*

  Using our terminology, the statement of [SGJ23, Theorem 3] claims the UC-security of OEKE-RO, while its proof is about the UC-security of OEKE-PRF; the latter requires an additional assumption on the underlying KA protocol (which is overlooked in the proof). However, it is never specified what this "additional assumption" is! As such, at least the security proof of [SGJ23, Theorem 3] seems incomplete.[4] There is also a separate issue that renders this result incorrect no matter whether OEKE-PRF or OEKE-RO is considered.

[4] Looking ahead, this "additional assumption" is what we call *psuedorandom non-malleability* for the underlying KA protocol. Furthermore, even OEKE-RO requires pseudorandom non-malleability in the general case (but not when instantiating OEKE-RO with Diffie-Hellman KA); that is, one cannot "get rid of the additional assumption" even if they use OEKE-RO. See Sect. 3 and Appx. B for a detailed discussion.

- [LLHG23, Theorem 2] proves that an instantiation of EKE (called 2DH-EKE therein) is *tightly* UC-secure under CDH if the underlying KA protocol is hashed *twin* Diffie-Hellman (where one party sends two group elements $g^{a_1}$ and $g^{a_2}$ and the other sends a single group element $g^b$, and the KA key is $H(g^{a_1 b}, g^{a_2 b})$; in the PAKE protocol the password is also included in the RO hash) and the underlying encryption scheme is IC. This can be viewed as a variant of the result in [DHP+18] (with greater computation and communication cost but tighter security bound), and has the same two weaknesses as [DHP+18].

- Finally, [BCP+23, Theorem 1] proves the UC-security of EKE using IC and a secure and pseudorandom KA protocol, with the session key being an RO hash of the KA key together with the PAKE transcript. This result has a flavor similar to [MRR20, Theorem 10] and [SGJ23, Theorem 2], but is less efficient (it uses IC rather than POPF or HIC). Close scrutiny shows that this result is also incorrect: we first show a flaw in a reduction in the hybrid proof very similar to the issue in [DHP+18, Theorem 6], and then give an actual attack that breaks the UC-security.
  [BCP+23, Theorem 2] proves the UC-security of OEKE-RO using IC and a secure and pseudorandom KA protocol, with the session key being an RO hash of the KA key together with the PAKE transcript. We will show that this security statement is also incorrect.

See Table 2 for a summary of existing UC security analyses of (O)EKE.

We can see that despite a large amount of works analyzing the UC-security of (O)EKE in various flavors, the state of art remains unsatisfactory. In particular, all existing analyses of EKE, except the one using IC and the specific hashed twin Diffie-Hellman KA, are flawed; and all existing analyses of OEKE are flawed. This naturally begs the question we ask in the title:

*Under what conditions is EKE (and its variants) actually secure?*

### 1.2 Our Contributions

In this work, we present a comprehensive and thorough study of the UC-security of EKE and its one-encryption variant. We consider the protocols instantiated with the *most efficient* encryption scheme, i.e., POPF; and the *most general* KA protocols, which allows us to obtain a wide range of protocols, including ones based on post-quantum assumptions. We pinpoint the exact properties the underlying KA protocol needs to satisfy, which (in addition to the standard notions of correctness and security) we call *strong pseudorandomness*, *pseudorandom non-malleability* and *collision resistance* — all of which are missing in existing works. Concretely, our contributions are as follows:

**Subtleties in (O)EKE analyses and flaws in existing works.** In Sect. 3 we point out several subtleties in the UC security analyses of (O)EKE, using IC and Diffie-Hellman KA (including its plain and hashed versions) as an example:[5]

- We formally prove that EKE with plain Diffie-Hellman is insecure, which implies that [MRR20, Theorem 10] and [SGJ23, Theorem 2] are false;

- We show that OEKE-PRF with plain Diffie-Hellman is generally insecure;

[5]The reason is that IC is the strongest encryption scheme, so if an (O)EKE instantiation using IC is insecure, then it is also insecure using HIC or POPF; for Diffie-Hellman KA, it is the KA protocol that has been analyzed the most in (O)EKE, and its simplicity allows us to illustrate our main points relatively easily.

| result | protocol analyzed | KA protocol | encryption scheme | note |
|---|---|---|---|---|
| [DHP+18, Theorem 6] | EKE | hashed Diffie-Hellman (PAKE transcript included in hash) | IC | proof flawed (Sect. 3.3) |
| [MRR20, Theorem 10] | EKE | general | POPF | result incorrect (Sects. 3.1 and 3.5, Appx. D) |
| [SGJ23, Theorem 2] | EKE | general | HIC | result incorrect (Sects. 3.1 and 3.5) |
| [SGJ23, Theorem 3] | OEKE-? | general | HIC | result ambiguous, unclear if OEKE-PRF or OEKE-RO either way result incorrect (Sect. 3.4) |
| [LLHG23, Theorem 2] | EKE | hashed twin Diffie-Hellman (password included in hash) | IC | |
| [BCP+23, Theorem 1] | EKE | general (PAKE transcript included in hash) | IC | result incorrect (Sect. 3.3, Appx. B) |
| [BCP+23, Theorem 2] | OEKE-RO | general (PAKE transcript included in hash) | IC | result incorrect (Appx. B) |
| our Thm. 5.4 | EKE | general | POPF | |
| our Thm. 5.6 | OEKE | general | POPF | |

Table 2: UC security analyses of (O)EKE. Flawed analyses are marked in grey. (The attacks in Sects. 3.1 and 3.5 were originally presented in a talk by Stanislaw Jarecki [Jar23], whereas those in other sections were discovered by us. See the corresponding sections for a more detailed explanation of the history of the attacks)

- We show that for EKE with hashed Diffie-Hellman, it is unlikely to have a tight reduction to either the CDH assumption or the Decisional Diffie-Hellman (DDH) assumption (for CDH the reduction loses a factor of $\Theta(q^2)$ where $q$ is the number of the adversary's RO queries, while for DDH the reduction loses a factor of $\Theta(q)$), and we pinpoint the exact places where the proofs of [BCP+23, Theorem 1] and [DHP+18, Theorem 6] (which do not have this security loss) break down.

All of the above show that the underlying KA protocol must satisfy the notion of *pseudorandom non-malleability*, which we then give a formal definition. (Plain Diffie-Hellman does not satisfy this property, and hashed Diffie-Hellman satisfies loose pseudorandom non-malleability under CDH or DDH.) Furthermore,

- We show that OEKE (including its PRF and RO variants) is insecure if the underlying (plain or hashed) Diffie-Hellman KA protocol allows the identity group element as a valid message. This implies that [SGJ23, Theorem 3], in addition to being ambiguous, is false no matter which interpretation we take.

This points to a separate property the underlying KA protocol needs to satisfy, which we name *collision resistance*.

Finally, we show another attack on EKE that works only if the underlying encryption scheme is HIC or POPF (rather than IC). Concretely,

- We show that EKE using HIC or POPF only realizes a relaxed UC PAKE functionality *no matter what the KA protocol is*, and in order to realize the standard functionality, the session key needs to be $\mathsf{PRF}_K(\phi')$, where $K$ is the KA key and $\phi'$ is the second PAKE message. We call this modified protocol EKE-PRF. This implies that [MRR20, Theorem 10] and [SGJ23, Theorem 2] are false for a reason different from the first bullet above.

In Appx. A we show some additional attacks that only work for some general underlying KA protocols that are not Diffie-Hellman. The main difference is that Diffie-Hellman is perfectly pseudorandom (i.e., the messages are uniform), and additional subtleties are involved if the KA protocol only has computational pseudorandomness. These attacks show that KA *strong pseudorandomness* is needed for the security of EKE, and [MRR20, Theorem 10] is false in yet another way; and pseudorandom non-malleability is needed even in OEKE-RO, and [BCP+23, Theorem 2] is false.

**POPF in the almost UC framework.** To analyze (O)EKE with more than just IC, we use the POPF concept defined by [MRR20]. At a high level, a POPF is a function family $\{F_\phi\}$ that is random everywhere, except that for adversarially chosen $\phi$, the adversary can program one input/output pair $(x, y)$ such that $F_\phi(x) = y$. (This can be viewed as a randomized version of IC, where $F_\phi(x) = y$ is analogous to $\mathcal{D}(x, \phi) = y$, but unlike IC where with overwhelming probability there is only a single $\phi$ such that $\mathcal{E}(x, y) = \phi$, the adversary can program $(x, y)$ such that $F_\phi(x) = y$ holds for multiple indices $\phi$. The "programmable-once" property is crucial for the simulator to unambiguously extract a password guess $x$ from PAKE message $\phi$.) However, the game-based POPF security definition of [MRR20, Definition 7] appears insufficient for fully proving the UC-security of EKE; in particular, [MRR20] mistakenly only considers attacks where a single party is corrupted, leaving out the case the most significant case of a man-in-the-middle adversary. To address the limitations of the game-based POPF definition, we sought to define a UC ideal functionality representing a POPF.

However, the *uncontrollable outputs* property of POPF turns out to be difficult to model with UC. It states that we can extract a specific input $x^*$ from any adversarially-chosen POPF index $\phi$, and the adversary's influence is concentrated at $x^*$ so that it "cannot control" the POPF evaluation

$F_\phi(x)$ for any $x \neq x^*$. The game-based POPF definition formalized this as requiring that $F_\phi(x)$ be a suitable input for any weak PRF, but this does not fit into the UC framework. Note that we cannot have the ideal functionality randomly sample $F_\phi(x)$ for $x \neq x^*$, as the proof for the 2-round Feistel network POPF [MRR20, Theorem 9] requires a loose reduction that makes a guess over the adversary's oracle queries, which is not allowed in a UC simulator.

We solve this by moving to *almost UC*, where instead of requiring that a POPF securely realize the POPF functionality, we only require that the POPF compose securely with any protocol built on top of the POPF functionality.[6] We can then make the ideal functionality have $F_\phi(x)$ chosen randomly for $x \neq x^*$, without forcing the POPF to have a perfect simulation. To show the viability of our almost UC POPF definition, we prove that the 2-round Feistel network satisfies our almost UC definition of POPF. When proving security for the composition of this POPF with the protocol built on top, the guessing required for uncontrollable outputs now appears only inside hybrids, and not the actual simulator.

**Analysis of (O)EKE using POPF and suitable KA protocols.** Equipped with our almost UC modeling of POPF, we now turn to our main contribution: a multifaceted analysis of the most efficient instantiations of both EKE and OEKE in the UC framework. As we have seen, there are a large number of subtleties in (O)EKE and their analysis, causing several flaws in existing works. These subtleties come from three dimensions (apart from the complexity of the UC PAKE model itself):

1. There seem to be significant confusion over how the PAKE session keys should be derived. For EKE, [MRR20, SGJ23] set the session key to be the "raw" KA key, [DHP+18, BCP+23] use an RO hash on the KA key together with the PAKE transcript, and [LLHG23] includes the password in the final hash. For OEKE, [SGJ23] is ambiguous about whether the session key should be derived using a PRF or an RO on the KA key, whereas [BCP+23] hashes the KA key together with the PAKE transcript. It has never been properly examined or explained what exact items (if any) need to be hashed, and why.[7]

2. It is also not entirely clear what the exact security requirements the underlying KA protocol needs to satisfy are. All existing general analyses of (O)EKE [MRR20, SGJ23, BCP+23] only require the KA protocol to be secure and pseudorandom, which is incorrect: both EKE and OEKE require some further non-standard properties.

3. Finally, the differences between HIC/POPF and IC — which lead to additional attacks on EKE instantiated with HIC/POPF — are also overlooked in existing works, causing incorrect security statements [MRR20, SGJ23].

In this work, we present thorough clarifications of *all* issues above, together with an analysis of the most efficient instantiations to date:

1. *"Minimal" output function*: We show that for EKE using POPF, the session key needs to be a PRF of the second PAKE message under the KA key, and no other items need to be included; if we use IC instead, outputting the "raw" KA key suffices (provided that the KA protocol

[6] This idea of only requiring it to preserve the security of protocols built on top is similar to security-preserving samplers defined by [AWZ23]. The goal there is quite different however: the elimination of ROs from a 1-round protocol.

[7] Throughout this paper, by default EKE and OEKE refers to their "raw" versions, i.e., the session key is the KA key (or in the case of OEKE, first half of the KA key); the variants with the PAKE transcript and/or the password included in the RO are discussed along the way. OEKE-PRF and OEKE-RO also refer to the versions where the PAKE transcript and the password are not included in the PRF/RO.

satisfies appropriate properties — see below). For OEKE, we show that the "raw" version is secure, and no items need to be hashed (again, provided that the KA protocol satisfies appropriate properties); this in particular covers both the PRF and RO variants.

2. *Exact KA properties*: We show that for EKE to be secure, the underlying KA protocol needs to satisfy what we call *strong pseudoranodomness* and *pseudorandom non-malleability*, in addition to the standard properties of correctness and security. As its name suggests, strong pseudorandomness is a strengthening of (standard) pseudorandomness. We give a concrete counterexample showing that strong pseudorandomness is not implied by properties considered in existing works. While pseudorandom non-malleability is implied by security *in the specific case of hashed Diffie-Hellman*, (1) the reduction to security is non-tight, which is overlooked in [DHP+18, BCP+23]; (2) the implication does not hold if the KA key is unhashed (i.e., if we use plain Diffie-Hellman), which is overlooked in [MRR20, SGJ23]; and (3) it is not implied by security and (strong) pseudorandomness in the general case, which is overlooked in [BCP+23]. As such, pseudorandom non-malleability needs to be presented as a separate KA property of its own.
   Regarding OEKE, we show that for OEKE to be secure the underlying KA protocol also needs to satisfy pseudorandom non-malleability, which is overlooked in [SGJ23, BCP+23]; plus another property called *collision resistance*, which is overlooked in [SGJ23]. (If we use OEKE-RO with the password included in the hash while deriving the authenticator — which is what [BCP+23] does — then collision resistance is not needed.)
   To the best of our knowledge, we are the first to formalize the KA notions of strong pseudorandomness, pseudorandom non-malleability and collision resistance, let alone to point out that they are necessary for the security of (O)EKE.

3. *Most efficient instantiations*: Our main results about both EKE and OEKE use POPF, which is the most efficient instantiation to date. This provides significant efficiency advantage over the traditional IC implementation, as POPF only requires 2 rounds of Feistel network (as opposed to 8 in IC).

Our goal is to not only present *what* is exactly needed, but also develop a thorough understanding of *why* exactly they are needed. To achieve this, we include a large number of comments and explanations along with our security proofs and attacks, as well as a comprehensive summary at the end of the paper (Sect. 6.2).

Our general, modular approach provides two advantages, one theoretical and one practical:

- Some existing works [DHP+18, LLHG23] only analyze EKE using the specific Diffie-Hellman KA (or some variants of it), which "hides" the exact requirements for the underlying KA protocol, since Diffie-Hellman has perfect pseudorandomness, and its second message does not depend on the first (i.e., the two messages can be sent simultaneously). As we will see — as early as in Sect. 2 (preliminaries) — these properties significantly simplify the analysis, and an analysis using general KA is much more subtle and complicated. This means that our general analysis forces us to carefully consider and pinpoint the exact properties the underlying KA needs to satisfy.

- Our analysis provides a general framework for instantiating (O)EKE with various KA protocols, including post-quantum ones such as Kyber. In other words, given our results, the (in)security of a specific (O)EKE instantiation reduces to whether the underlying KA protocol satisfies some properties, which is much easier to verify.

# 2 Preliminaries

For an (efficiently samplable) set $S$, we write $x \leftarrow S$ for sampling an element from $S$ according to the uniform distribution. For an algorithm $\mathcal{A}$, we write $y \leftarrow \mathcal{A}(x; r)$ for running $\mathcal{A}$ on input $x$ and randomness $r$, and obtaining $\mathcal{A}$'s output $y$; if $\mathcal{A}$ is deterministic, we instead write $y := \mathcal{A}(x)$. We use $\kappa$ to denote the security parameter. For group operations, we use the multiplicative notation.

**UC conventions.** We assume w.l.o.g. that the session ID always includes the names of the two parties. Furthermore, for ideal objects such as RO and IC, we always assume that the session ID is part of the input (for IC, it is part of the key, i.e., the protocol messages are in the form of $\mathcal{E}(\mathsf{sid} \parallel \mathsf{pw}, A)$) and do not explicitly write them. This is for brevity only; we stress that in actual protocols the session ID should be included to achieve domain separation.

## 2.1 (Unauthenticated) Key Agreement Protocols

A 2-round *Key Agreement (KA)* protocol (henceforth KA protocol) consists of the following (deterministic) algorithms:

- $\mathsf{msg}_1(a) = A \in \mathcal{M}_1$: protocol-message function for party $P$
- $\mathsf{msg}_2(b, A) = B \in \mathcal{M}_2$: protocol-message function for party $P'$
- $\mathsf{key}_1(a, B) = K \in \mathcal{K}$: output function for party $P$
- $\mathsf{key}_2(b, A) = K' \in \mathcal{K}$: output function for party $P'$

In a standard execution of the protocol, party $P$ samples randomness $a \leftarrow \mathcal{R}$ and sends $A := \mathsf{msg}_1(a)$ to party $P'$. $P'$ then samples $b \leftarrow \mathcal{R}$, sends $B := \mathsf{msg}_2(b, A)$ to $P$, and outputs $K' := \mathsf{key}_2(b, A)$. Upon receiving $B$, $P$ outputs $K := \mathsf{key}_1(a, B)$.

**Definition 2.1.** *A KA protocol is **correct** if*

$$\mathsf{key}_1(a, \mathsf{msg}_2(b, \mathsf{msg}_1(a))) = \mathsf{key}_2(b, \mathsf{msg}_1(a))$$

*with overwhelming probability when $a, b \leftarrow \mathcal{R}$.*

**Security.**

**Definition 2.2.** *A KA protocol is **secure** if the following two distributions are indistinguishable:*

| | |
|---|---|
| $a \leftarrow \mathcal{R}$ | $a \leftarrow \mathcal{R}$ |
| $b \leftarrow \mathcal{R}$ | $b \leftarrow \mathcal{R}$ |
| $A := \mathsf{msg}_1(a)$ | $A := \mathsf{msg}_1(a)$ |
| $B := \mathsf{msg}_2(b, A)$ | $B := \mathsf{msg}_2(b, A)$ |
| $K := \mathsf{key}_1(a, B)$ | $K \leftarrow \mathcal{K}$ |
| output $(A, B, K)$ | output $(A, B, K)$ |

**Pseudorandomness.** We also consider the notion of pseudorandomness in which the protocol messages are also indistinguishable from random.

**Definition 2.3.** *A KA protocol has* ***pseudorandom first message****, or* ***first pseudorandomness****, if the following two distributions are indistinguishable:*

| | |
|---|---|
| $a \leftarrow \mathcal{R}$<br>$A := \mathsf{msg}_1(a)$<br>output $A$ | $A \leftarrow \mathcal{M}_1$<br>output $A$ |

**Definition 2.4.** *A KA protocol has* ***pseudorandom second message****, or* ***second pseudorandomness****, if the following two distributions are indistinguishable:*

| | |
|---|---|
| $a \leftarrow \mathcal{R}$<br>$A := \mathsf{msg}_1(a)$<br>$b \leftarrow \mathcal{R}$<br>$B := \mathsf{msg}_2(b, A)$<br>output $(A, B)$ | $a \leftarrow \mathcal{R}$<br>$A := \mathsf{msg}_1(a)$<br><br>$B \leftarrow \mathcal{M}_2$<br>output $(A, B)$ |

The two pseudorandomness properties combined imply that the *joint distribution* of the two KA messages are indistinguishable from random.

Existing works on EKE use different terms for KA pseudorandomness and security, some of which are presented in the terms of KEM. See Table 3 for a comparison.

| | first KA message / KEM public key pseudorandomness | second KA message / KEM ciphertext pseudorandomness | KA/KEM key psuedorandomness |
|---|---|---|---|
| [MRR20] | KA pseudorandomness | | KA security |
| [SGJ23, Section 2.1] | KA random message | | KA security |
| [SGJ23, Section 2.2] | KEM uniform public key | KEM anonymity[8] | KEM IND-security |
| [BCP+23] | KEM fuzziness | KEM anonymity | KEM indistingushability |
| this work | KA first pseudorandomness | KA second pseudorandomness | KA security |

Table 3: Terminologies for KA pseudorandomness and security

**Strong pseudorandomness.** In the security analysis for EKE we need a stronger pseudorandomness property, which says that the second message is pseudorandom *even given the key derivation function used on it*:

**Definition 2.5.** *A KA protocol has* ***strong pseudorandom second message****, or* ***strong second***

[8] [SGJ23, Section 2.2] uses "KEM anonymity" to refer to a slightly weaker property, where two KEM ciphertexts on two randomly generated public keys are indistinguishable. ("KEM anonymity" in [BCP+23] requires the KEM ciphertext to be pseudorandom over the ciphertext space, i.e., it is equivalent to our KA second pseudorandomness.)

***pseudorandomness**, if the following two distributions are indistinguishable:*

| | |
|---|---|
| $a \leftarrow \mathcal{R}$ | $a \leftarrow \mathcal{R}$ |
| $A := \mathsf{msg}_1(a)$ | $A := \mathsf{msg}_1(a)$ |
| $b \leftarrow \mathcal{R}$ | |
| $B := \mathsf{msg}_2(b, A)$ | $B \leftarrow \mathcal{M}_2$ |
| $K := \mathsf{key}_1(a, B)$ | $K := \mathsf{key}_1(a, B)$ |
| output $(A, B, K)$ | output $(A, B, K)$ |

Henceforth we may call the combination of first and second pseudorandomness *pseudorandomness*, and the combination of first and strong second pseudorandomness *strong pseudorandomness*.

At first glance, strong pseudorandomness might appear to be implied by the standard notions of security and pseudorandomness. However, in Appx. A we show that this is not the case. We also present an attack demonstrating why strong pseudorandomness is necessary for the security of EKE (but not OEKE). This distinction is overlooked in all existing works that use general 2-round KAs.

**Example: Diffie-Hellman.** The *plain (resp. hashed) Diffie-Hellman KA* is defined as follows: fix some cyclic group $(\mathbb{G}, p, g)$, and the relevant spaces are $\mathcal{R} = \mathbb{Z}_p$, $\mathcal{M}_1 = \mathcal{M}_2 = \mathbb{G}$, and $\mathcal{K} = \mathbb{G}$ (resp. $\mathcal{K} = \{0,1\}^\kappa$). The algorithms are

$$\begin{aligned} \mathsf{msg}_1(a) &= g^a \\ \mathsf{msg}_2(b, A) &= g^b \\ \mathsf{key}_1(a, B) &= B^a \text{ (resp. } H(B^a)) \\ \mathsf{key}_2(b, A) &= A^b \text{ (resp. } H(A^b)) \end{aligned}$$

where $H : \mathbb{G} \to \{0,1\}^\kappa$ is a hash function (usually modeled as an RO). Note that the output of $\mathsf{msg}_2(b, A)$ does not depend on $A$, so this protocol can be executed in 1 simultaneous round; however, we present it as $P'$ waits for message from $P$, in order to fit the general description of 2-round KA protocols.

It is well-known that plain Diffie-Hellman satisfies computational security under the DDH assumption, and its hashed variant satisfies computational security under the CDH assumption, the DDH assumption, or the Gap Diffie-Hellman (GDH) assumption, with a security loss of $q$ (the number of the adversary's $H$ queries) under CDH and no security loss under DDH or GDH. Both plain Diffie-Hellman and hashed Diffie-Hellman satisfy perfect correctness and perfect strong psuedorandomness.

**Example: Kyber.** *Kyber* [SAB+22] is a post-quantum KEM currently under consideration for standardization by NIST. Without going into much detail, Kyber starts with a public-key encryption scheme based on module-LWE PKE, then applies the Fujisaki-Okamoto (FO) transform [FO99] to achieve KEM with CCA-security. The underlying public-key encryption has public keys and ciphertexts that can be thought of vectors over $\mathbb{Z}_q$ for $q = 3329$, and correctness and CPA-security hold under the module-LWE assumption. Unfortunately, Kyber includes an optimization to compress the ciphertexts by rounding off some of the least significant bits, and this causes a small bias in the ciphertext distribution because $q$ is not a power of two. Therefore, we must consider Kyber without this compression optimization. Without it, Kyber's public-key encryption scheme has pseudorandom public keys and ciphertexts [BCP+23].

We now describe the Kyber KEM, though we write it as a KA instead. This version is somewhat

simplified compared to actual Kyber, as some of the hashes have been rearranged to make the presentation simpler by removing some of its optimizations. However, the modified hash calls are indifferentiable from the originals, so this should make no difference to the security arguments.[9] The algorithms are

$$\begin{aligned}
\mathsf{msg}_1(a) &= \mathsf{Kyber.KeyGen}(a) \\
\mathsf{msg}_2(b, A) &= \mathsf{Kyber.Enc}(A, b; H(b, A)) \\
\mathsf{key}_2(b, A) &= H'(b, A, \mathsf{msg}_2(b, A)) \\
\mathsf{key}_1(a, B) &= \begin{cases} H'(\mathsf{Kyber.Dec}(a, B), A, B) & \text{if } B = \mathsf{msg}_2(\mathsf{Kyber.Dec}(a, B), A) \\ H'(a, B) & \text{otherwise} \end{cases}
\end{aligned}$$

where $H'$ is a hash function onto $\{0,1\}^\kappa$ (usually modeled as an RO). (Here, $\mathsf{Kyber.Enc}(A, m; r)$ means to encrypt message $m$ under public key $A$ using randomness $r$.)

For an argument why Kyber satisfies various properties we need, see Appx. C.

## 2.2 UC PAKE Functionality

We recall the standard UC PAKE functionality [CHK+05] in Figure 3. Below we briefly describe the idea behind the functionality; for a more detailed explanation, see [RX23, Section 2.2].

- On input $(\mathsf{NewSession}, \mathsf{sid}, P, P', \mathsf{pw}, \mathsf{role})$ from $P$, send $(\mathsf{NewSession}, \mathsf{sid}, P, P', \mathsf{role})$ to $\mathcal{S}$. Furthermore, if this is the first NewSession message for *sid*, or this is the second NewSession message for sid and there is a record $\langle P', P, \cdot \rangle$, then record $\langle P, P', \mathsf{pw} \rangle$ and mark it fresh.
- On $(\mathsf{TestPwd}, \mathsf{sid}, P, \mathsf{pw}^*)$ from $\mathcal{S}$, if there is a record $\langle P, P', \mathsf{pw} \rangle$ marked fresh, then do:
  - If $\mathsf{pw}^* = \mathsf{pw}$, then mark the record compromised and send "correct guess" to $\mathcal{S}$.
  - If $\mathsf{pw}^* \neq \mathsf{pw}$, then mark the record interrupted and send "wrong guess" to $\mathcal{S}$.
- On $(\mathsf{NewKey}, \mathsf{sid}, P, K^* \in \{0,1\}^\kappa)$ from $\mathcal{S}$, if there is a record $\langle P, P', \mathsf{pw} \rangle$, and this is the first NewKey message for *sid* and $P$, then output $(\mathsf{sid}, K)$ to $P$, where $K$ is defined as follows:
  - If the record is compromised, or either $P$ or $P'$ is corrupted, then set $K := K^*$.
  - If the record is fresh, a key $(\mathsf{sid}, K')$ has been output to $P'$, at which time there was a record $\langle P', P, \mathsf{pw} \rangle$ marked fresh, then set $K := K'$.
  - Otherwise sample $K \leftarrow \{0,1\}^\kappa$.

  Finally, mark the record completed.

Figure 3: UC PAKE functionality $\mathcal{F}_{\text{PAKE}}$

The functionality involves two parties, $P$ with password pw and $P'$ with password pw′. Each execution of the protocol incurs a session for $P$ (i.e., a session initiated by $P$ where $P$ eventually outputs a key) and a session for $P'$, both of which have a corresponding record marked with a state:

- When a session is established (by a NewSession command), it is marked fresh, and will remain fresh unless and until the (ideal) adversary attacks the session.

[9] There are multiple versions of Kyber; our description follows the most recent draft submission (version 3).

- The adversary may attack a fresh $P$ session by sending a TestPwd command for $P$, on a *password guess* $\mathsf{pw}^*$. This models an *online guessing attack* in the real world, where the adversary runs the algorithm of $P'$ on a password guess $\mathsf{pw}^*$ and communicates with $P$. If $\mathsf{pw}^* = \mathsf{pw}$, this is a successful attack, and the session is marked compromised; otherwise the session is marked interrupted. (Symmetrically, the adversary may attack the $P'$ session by sending a TestPwd command for $P'$.) Note that once a session becomes compromised or interrupted, it can never return to fresh; this in particular means that TestPwd can be run only once on any specific session.

- The adversary may end a session by sending a NewKey command, and the session will output a key depending on the states of this session and its counter session. After that, the session is marked completed (so that TestPwd cannot be sent after the session ends).

It is subtle and critical to our later sections how exactly a session key is determined, so let us explain the three cases under NewKey further:

- The "normal" case is that both the $P$ session and the $P'$ session are fresh, and $\mathsf{pw} = \mathsf{pw}'$. This models a correct protocol execution in the real world, where the adversary does not interfere and the two parties' passwords match. Say the $P'$ session ends first; then $P'$ outputs a random session key $K'$ (the third case under NewKey), and when the $P$ session ends, $P$ outputs session key $K = K'$ (the second case).

- If both the $P$ session and the $P'$ session are fresh, but $\mathsf{pw} \neq \mathsf{pw}'$, then this corresponds to an incorrect protocol execution where the adversary does not interfere but the two parties' passwords do not match. In this case $P$ and $P'$ output independent random keys (the third case).

- If the $P$ session is compromised, it models the real-world scenario where the adversary has successfully performed an online guessing attack on $P$. In this case all security guarantees are lost, so we might as well let the adversary choose the session key for $P$ (the first case). The same goes for the $P'$ session (same below).

- If the $P$ session is interrupted, it models the real-world scenario where the adversary has performed an unsuccessful online guessing attack on $P$. This means that the adversary should have no information about the session key of $P$; furthermore, the session key of $P$ should also be independent of the session key of $P'$. So we let $P$ output a random session key (the third case).

- Finally, if the $P$ session is fresh but its counter session $P'$ has been attacked (either compromised or interrupted), then again the adversary should have no information about the session key of $P$ (because the $P$ session is fresh), and the session key of $P$ should also be independent of the session key of $P'$ (because the $P'$ session is attacked, so $P'$ should output a session key that is either set by the adversary or independent of everything else). So we also let $P$ output a random session key (the third case).

We stress that if a party's session key is random (the first three cases), *the adversary never gains any information about it.* For example, if the $P$ session is fresh and the $P'$ session is compromised, then the session key of $P$ is still independent of the adversary's view (even though the adversary fully controls the session key of $P'$). The above holds even if the adversary learns the party's password after the session ends with a random session key; in this case the session is already marked completed, so the adversary cannot send TestPwd even if it gets to know the password later on.

# 3 Subtleties in (O)EKE Security Analysis

In this section, we show that a number of (O)EKE instantiations, including the ones claimed secure in [MRR20] and [SGJ23], are actually insecure. We also show the flaws in the proofs of [BCP+23, Theorem 1] and [DHP+18, Theorem 6]. (In fact the instantiation in [BCP+23, Theorem 1] is also insecure due to a separate issue, but this is harder to see and is shown in Appx. B.) Since an IC is also a POPF and an HIC, in Sects. 3.1 to 3.4 we assume that the encryption scheme used in EKE is an IC and our attacks immediately apply to instantiations using POPF or HIC. The attack in Sect. 3.5 applies to EKE using POPF/HIC only.

All attacks in this section (except for Sect. 3.4) assume the passwords of party $P$ and party $P'$ match, which we call $\mathsf{pw}$. We stress that in the proofs one must also consider the case where the two parties' passwords are different.

## 3.1 EKE with Plain Diffie-Hellman Is Insecure

We first consider the EKE instantiation where the underlying KA protocol is plain Diffie-Hellman; that is, $P$ sends $\mathcal{E}(\mathsf{pw}, g^a)$ to $P'$ and $P'$ sends $\mathcal{E}(\mathsf{pw}, g^b)$ to $P$, and both parties decrypt each other's message and agree upon the session key $g^{ab}$. This protocol is insecure due to the following man-in-the-middle attack: an adversary can pass the $P$-to-$P'$ message $\mathcal{E}(\mathsf{pw}, g^a)$ without modification, then guess $\mathsf{pw}$ and replace the $P'$-to-$P$ message $\mathcal{E}(\mathsf{pw}, g^b)$ with $\mathcal{E}(\mathsf{pw}, g^{2b})$ (by decrypting the message and obtaining $g^b$, then encrypting $(g^b)^2$ under $\mathsf{pw}$). Assuming the password guess is correct, $P$ outputs $K = g^{2ab}$ and $P'$ outputs $K' = g^{ab}$, so $K = (K')^2$. In other words, an adversary that does not attack the $P'$ session but successfully attacks the $P$ session, causes the session keys of $P$ and $P'$ to be correlated — which is not allowed in a UC-secure PAKE.

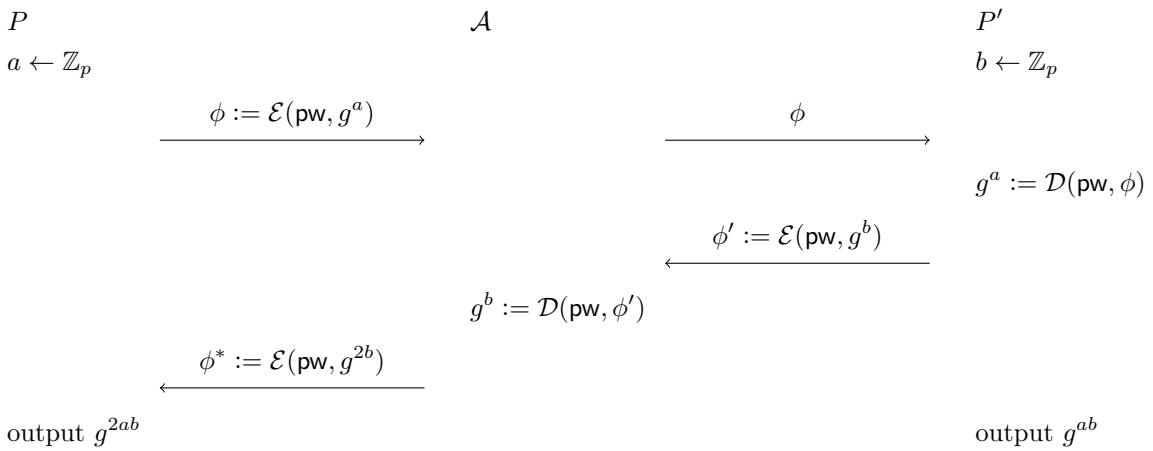


Figure 4: Attack on EKE with plain Diffie-Hellman. $\mathcal{A}$ only guesses $\mathsf{pw}$ in the second round

Indeed, the UC PAKE functionality $\mathcal{F}_{\text{PAKE}}$ guarantees that if the $P'$ session is not attacked but its counter-session was (successfully or unsuccessfully) attacked, then the $P'$ session is $\mathsf{fresh}$ and *the session key of $P'$ is independent of everything else* (see the last bullet at the end of Sect. 2.2). This in particular means that the protocol above violates UC-security for PAKE: when the adversary $\mathcal{A}$ passes $\mathcal{E}(\mathsf{pw}, g^a)$ to $P'$, the UC simulator $\mathcal{S}$ does not know $\mathsf{pw}$ and has to let $P'$ output a session key by sending a $(\mathsf{NewKey}, \mathsf{sid}, P', \star)$ message to $\mathcal{F}_{\text{PAKE}}$, causing $P'$ to output a random session key $K'$ (independent of the view of $\mathcal{S}$). Later, when $\mathcal{A}$ sends $\mathcal{E}(\mathsf{pw}, g^{2b})$ to $P$, $\mathcal{S}$ can extract $\mathsf{pw}$ and

compromise the $P$ session by sending a correct $\mathsf{TestPwd}$ message, but $K'$ is still independent of the view of $\mathcal{S}$, so $\mathcal{S}$ cannot make $K'$ and the session key of $P$ correlated.

It is not hard to turn the above observation into a formal proof of UC-insecurity:

**Theorem 3.1.** *EKE with the plain Diffie-Hellman KA does not UC-realize $\mathcal{F}_{\text{PAKE}}$ in the $\mathcal{F}_{\text{IC}}$-hybrid world.*

*Proof.* We assume the password dictionary $\mathsf{Dict}$ is a priori fixed and known to the simulator; this only makes the simulation potentially easier. There is no restriction on $\mathsf{Dict}$ except that $|\mathsf{Dict}| \geq 2$. Suppose there is a simulator $\mathcal{S}$ that generates an indistinguishable view for any PPT environment. Consider the environment $\mathcal{Z}$ in Figure 5 (for brevity, we omit $\mathsf{sid}$ in parties' messages and outputs below).

1. Sample $\mathsf{pw} \leftarrow \mathsf{Dict}$, and send $(\mathsf{NewSession}, \mathsf{sid}, P, P', \mathsf{pw})$ to $P$ and $(\mathsf{NewSession}, \mathsf{sid}, P', P, \mathsf{pw})$ to $P'$. // let $P$ and $P'$ run the protocol on the same password $\mathsf{pw}$
2. On $\phi$ from $P$, instruct the adversary to send $\phi$ to $P'$. Observe the output of $P'$, $K'$. // pass the $P$-to-$P'$ message without any modification
3. On $\phi'$ from $P'$, query $B := \mathcal{D}(\mathsf{pw}, \phi')$ and then $\phi^* := \mathcal{E}(\mathsf{pw}, B^2)$, and instruct the adversary to send $\phi^*$ to $P$. Observe the output of $P$, $K$. // replace the $P'$-to-$P$ message $\mathcal{E}(\mathsf{pw}, g^b)$ with $\mathcal{E}(\mathsf{pw}, g^{2b})$
4. Output 1 if $K = (K')^2$, and 0 otherwise. // guess "real world" iff $K = (K')^2$

Figure 5: Environment for EKE with plain Diffie-Hellman

In the real world, let $a$ be the randomness of $P$, $A = g^a$, and $b$ be the randomness of $P'$. Then $K' = A^b = g^{ab}$ and $K = (B^2)^a = g^{2ab}$, so $K = (K')^2$ and $\mathcal{Z}$ outputs 1 with probability 1.

In the ideal world, let $\mathsf{Compromise}$ be the event that the $P'$ session is $\mathsf{compromised}$ before it outputs $K'$. This happens if and only if $\mathcal{S}$ sends $(\mathsf{TestPwd}, \mathsf{sid}, P', \mathsf{pw})$ (i.e., making a correct password guess for $P'$) before $P'$ outputs. The crucial observation is that *at the end of step 2,* the view of $\mathcal{S}$ only consists of $(\mathsf{NewSession}, \mathsf{sid}, P, P')$ and $(\mathsf{NewSession}, \mathsf{sid}, P', P)$[10], so $\mathsf{pw}$ *is independent of the view of* $\mathcal{S}$. Thus,

$$\Pr[\mathsf{Compromise}] \leq \frac{1}{|\mathsf{Dict}|}.$$

Now assume that $\mathsf{Compromise}$ does not happen; in other words, when $\mathcal{S}$ lets $P'$ output $K'$ via a $\mathsf{NewKey}$ command to $\mathcal{F}_{\text{PAKE}}$, the $P'$ session is $\mathsf{fresh}$ or $\mathsf{interrupted}$. Either way, $\mathcal{F}_{\text{PAKE}}$ samples $K' \leftarrow \{0,1\}^\kappa$. In the rest of the experiment, the view of $\mathcal{S}$ consists of $\mathsf{pw}$ from IC queries, which is independent of $K'$, so $K'$ *is independent of the view of* $\mathcal{S}$ *throughout the experiment.* Next, consider the state of the $P$ session before it outputs $K$:

- If it is $\mathsf{fresh}$, and the $P'$ session was also $\mathsf{fresh}$ before $P'$ outputs $K'$, then $\mathcal{F}_{\text{PAKE}}$ enters the second case under $\mathsf{NewKey}$, so $K = K'$. Thus, $K = (K')^2$ iff $K' = e$ (the identity group element), which happens with probability $1/p$;
- If it is $\mathsf{interrupted}$, or it is $\mathsf{fresh}$ and the $P'$ session was $\mathsf{interrupted}$ before $P'$ outputs $K'$, then

[10] Formally it also consists of $\phi$ which is copied from the message simulated by $\mathcal{S}$ itself. Below we omit such messages for readability.

$\mathcal{F}_{\text{PAKE}}$ enters the third case under NewKey, so $K \leftarrow \mathbb{G}$ (independent of $K'$) and the probability that $K = (K')^2$ is $1/p$;

- If it is compromised (note that at this time $\mathcal{S}$ knows pw from IC queries, so it is able to compromise the $P$ session), then $\mathcal{F}_{\text{PAKE}}$ enters the first case under NewKey, so $K$ is set by $\mathcal{S}$. However, as we have just argued, $K'$ is a random element of $\mathbb{G}$ in the view of $\mathcal{S}$, so the probability that $K = (K')^2$ is $1/p$.

We conclude that as long as Compromise does not happen, the probability that $K = (K')^2$ — in other words, the probability that $\mathcal{Z}$ outputs 1 — is $1/p$. Overall, the probability that $\mathcal{Z}$ outputs 1 in the ideal world is at most

$$\Pr[\mathsf{Compromise}] + \frac{1}{p} \leq \frac{1}{|\mathsf{Dict}|} + \frac{1}{p},$$

so the distinguishing advantage of $\mathcal{Z}$ is at least

$$1 - \frac{1}{|\mathsf{Dict}|} - \frac{1}{p},$$

which is non-negligible since $|\mathsf{Dict}| \geq 2$. Thus, such a "successful" simulator $\mathcal{S}$ does not exist, which concludes the proof. □

**Remark 3.2.** *We note that the proof of Thm. 3.1 does not rely on the simulator $\mathcal{S}$ being PPT; in other words, the failure of the simulator is "statistical" and even a computationally unbounded simulator still cannot generate an indistinguishable view for our environment $\mathcal{Z}$.*

The above shows that [MRR20, Theorem 10] and [SGJ23, Theorem 2] are false, since both theorems only require security and pseudorandomness of the underlying KA protocol, which are satisfied by plain Diffie-Hellman.

**Remark 3.3.** *The attack above does not seem to affect the game-based security of EKE with plain Diffie-Hellman. Roughly speaking, in the game-based security definition, once the adversary guesses the correct password all security guarantees are lost; whereas in UC-security* the simulator needs to continue simulating the rest of the protocol (including the session key) even after the adversary guesses the correct password — *which is exactly where the simulator in EKE with plain Diffie-Hellman fails. This gap between the game-based security and UC-security for PAKE repeatedly appears in various protocols; as an early example, the Katz-Ostrovsky-Yung protocol [KOY01] is game-based secure but not UC-secure for exactly this reason (see [CHK+05, Section 2.3] for a detailed discussion). Since game-based security is not the focus of this work, we do not explore this topic further.*

**History of the attack.** We stress that the flaws above were not discovered by us. First, [SGJ23] seems to have pointed out that [MRR20, Theorem 10] is false ("[W]e think that it is unlikely that EKE can provably realize UC PAKE based on the POPF properties alone"), and [SGJ23] is right that some form of non-malleability is missing in [MRR20, Theorem 10]. However, [SGJ23] appears to believe that the non-malleability property lies in the *encryption scheme*, and using a stronger encryption scheme such as HIC resolves the issue; while in fact non-malleability is a property that must be satisfied by the underlying *KA protocol*, and as long as the KA protocol is plain Diffie-Hellman, the issue remains no matter what encryption scheme is used. In other words, [SGJ23] noticed that [MRR20, Theorem 10] is false but identified the reason incorrectly, which

explains why [SGJ23, Theorem 2] is also false.

Second, in the talk for the [SGJ23] paper [Jar23], the presenter Stanislaw Jarecki pointed out their own mistake and mentioned the attack on EKE with plain Diffie-Hellman, which we repeat here. According to [Jar23], the attack was found in a follow-up study (which we are not able to identify). As such, the credit belongs to [Jar23] and the follow-up work. However, the ePrint version of [SGJ23] has not been updated accordingly after the talk, and to the best of our knowledge, we are the first to present this attack in written form. Furthermore, the formal proof of UC-insecurity (Thm. 3.1) is our work.

### 3.2 OEKE-PRF with Plain Diffie-Hellman Is Not Necessarily Secure

We now proceed to show that the OEKE-PRF protocol is also not necessarily UC-secure if we use plain Diffie-Hellman as the underlying KA protocol. Recall that OEKE-PRF saves one IC operation: $P$ sends $\mathcal{E}(\mathsf{pw}, g^a)$ to $P'$, $P'$ samples integer $b$ and computes its KA key $K' = (g^a)^b = g^{ab}$ and PAKE session key $\mathsf{PRF}_{K'}(0)$, and finally sends $g^b$ together with $\tau' = \mathsf{PRF}_{K'}(1)$ to $P$. $P$ then computes $K = g^{ab}$ as $(g^b)^a$, and checks if $\tau' = \mathsf{PRF}_K(1)$ (if so, $P$ outputs $\mathsf{PRF}_K(0)$ as its session key; otherwise $P$ outputs a random session key).

Consider the following adversary that attacks this protocol: it passes the $P$-to-$P'$ message $\mathcal{E}(\mathsf{pw}, g^a)$ without modification, causing $P'$ to output $\mathsf{PRF}_{g^{ab}}(0)$. If $\mathsf{PRF}$ is such that $\mathsf{PRF}_{k^2}(x)$ is predictable from $\mathsf{PRF}_k(x)$ for a random $k \leftarrow \mathbb{G}$,[11] then the adversary, upon seeing $B = g^b$ and $\tau' = \mathsf{PRF}_{g^{ab}}(1)$ from $P'$, can send $B^* = B^2$ and $\tau^* = \mathsf{PRF}_{g^{2ab}}(1)$ to $P$; the check of $P$ will pass and $P$ will output $\mathsf{PRF}_{g^{2ab}}(0)$ — again, the session keys of $P$ and $P'$ are correlated. Note that unlike the attack on EKE in Sect. 3.1, here the adversary does not even need to know the password.

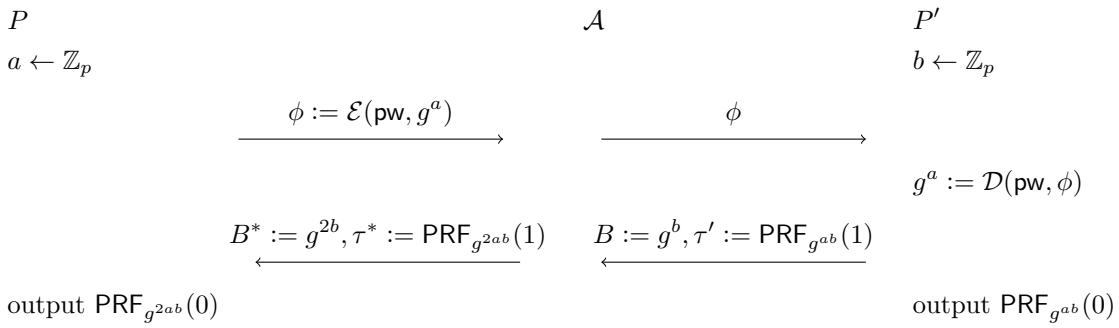


Figure 6: Attack on OEKE-PRF with plain Diffie-Hellman, assuming $\mathsf{PRF}_{k^2}(x)$ is predictable from $\mathsf{PRF}_k(x)$. $\mathcal{A}$ does not need to know $\mathsf{pw}$

The above shows that OEKE-PRF with a general PRF is insecure when instantiated with plain Diffie-Hellman; rather, the PRF has to satisfy the requirement that $\mathsf{PRF}_{k^2}(x)$ cannot be predicted from $\mathsf{PRF}_k(x)$. This condition is trivially met if $\mathsf{PRF}_k(x) = H(k, x)$ (where $H$ is an RO), i.e., OEKE-RO is not subject to this attack.

[11] Note that the security definition of PRF requires that the key be chosen at random, and says nothing about the PRF's behavior under two correlated keys.

### 3.3 Further Subtleties in EKE with Hashed Diffie-Hellman under CDH and DDH

The issues above appear to go away if we replace the output key $g^{ab}$ with $H(g^{ab})$, where $H$ is a hash function (potentially an RO). Indeed, since hashed Diffie-Hellman is secure under CDH/DDH in the ROM, one might conjecture that CDH/DDH suffices for the UC-security of EKE with this KA protocol. While this is true, close scrutiny reveals that there are further subtleties involving the *tightness* of the security analysis.

Consider a generalization of the attack on EKE in Sect. 3.1: the adversary (that correctly guesses $\mathsf{pw}$) passes $\phi = \mathcal{E}(\mathsf{pw}, g^a)$ from $P$ to $P'$ (so the session key of $P'$ is $H(g^{ab})$), and replaces the $P'$-to-$P$ message $\phi' = \mathcal{E}(\mathsf{pw}, g^b)$ with $(\phi')^* = \mathcal{E}(\mathsf{pw}, X)$, where $X \neq g^b$ is a group element of the adversary's choice. Then the session key of $P$ is $H(X^a)$. The UC-security of the protocol requires $H(g^{ab})$ to be pseudorandom even given $H(X^a)$ (again, because in the ideal world the simulator cannot compromise the $P'$ session, so the session key of $P'$, $H(g^{ab})$, is independent of everything else). In other words, UC-security implies the hardness of following problem: given $g^a, g^b$ (where $a, b \leftarrow \mathbb{Z}_p$), the adversary outputs a group element $X \neq g^b$ and receives $H(X^a)$, and needs to distinguish $H(g^{ab})$ from a random string.

This is the Oracle Diffie-Hellman (ODH) assumption, except that the oracle $H_a(\cdot)$ can be queried only once. (ODH says that given $g^a, g^b$ and access to $H_a(\cdot)$ which on input $X \neq g^b$ outputs $H(X^a)$, it is hard to distinguish $H(g^{ab})$ from random.) Henceforth we call this assumption *1-query ODH*. While 1-query ODH is equivalent to CDH in the ROM (i.e., assuming that $H$ is an RO), if DDH is hard the reduction to CDH loses a factor of $\Theta(q^2)$ where $q$ is the number of the adversary's $H$ queries. Indeed, an adversary on $g^a, g^b$ can sample a random integer $r \leftarrow \mathbb{Z}_p$, make $q$ queries to $H$ including $h_1 := H(g^{ar})$, output $X = g^r$ and receive $h_2$, check if $h_1 = h_2$, and abort if not. To simulate this, the reduction to CDH must make a guess on which $H$ query is $g^{ar}$ and lose a factor of $\Theta(q)$, since it only sees $g^a$ and $X = g^r$. Next, assuming the guess is correct, this essentially becomes reducing the security of hashed Diffie-Hellman to CDH — where the reduction needs to make a second guess on which $H$ query is $g^{ab}$, losing another factor of $\Theta(q)$.

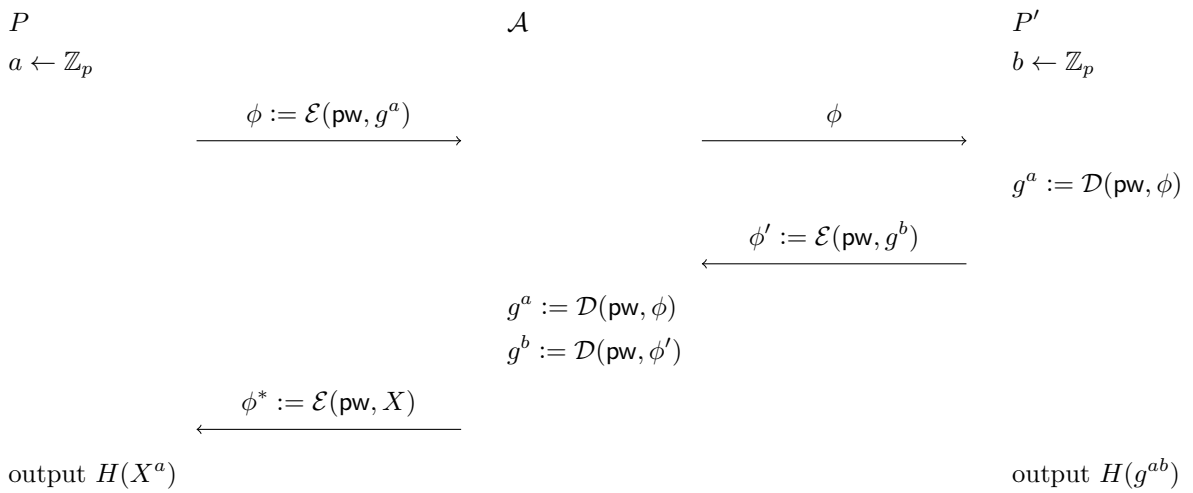


Figure 7: Attack on EKE with hashed Diffie-Hellman. $\mathcal{A}$ only guesses $\mathsf{pw}$ in the second round. UC-security requires $H(g^{ab})$ to be pseudorandom even given $H(X^a)$, which is 1-query ODH. Assuming $X = g^r$, $\mathcal{A}$ can make $q$ queries to $H$ including $H((g^a)^r)$, and $\mathcal{Z}$ can check for consistency with the output of $P$; the reduction to CDH/DDH needs to make a guess on which $H$ query is $H(g^{ar})$

The above shows that if we instantiate EKE with hashed Diffie-Hellman, there is a quadratic security loss while reducing to the CDH assumption. This was first briefly observed in [LLHG23] (and in fact is the starting point of that paper); however, [LLHG23] presents the reduction having to guess the $H(g^{ab})$ query and having to guess the $H(g^{ar})$ query as two *separate* issues, while in fact both of them can appear in the *same* session, causing a quadratic loss.

We note that if the reduction has access to a DH oracle, then both guesses can be replaced by going over all $H$ queries and checking which one is the "right" query; in other words, tight security can be achieved if we reduce to the GDH assumption. Furthermore, since hashed Diffie-Hellman is tightly secure under the DDH assumption, a reduction to DDH only needs to make the first guess, which incurs a $\Theta(q)$ security loss.[12]

| reduce to... | first guess | second guess | overall loss |
|---|---|---|---|
| CDH | ✓ | ✓ | $\Theta(q^2)$ |
| DDH | ✓ | | $\Theta(q)$ |
| GDH | | | none |

Table 4: Security loss of reducing 1-query ODH to various assumptions

The issue above persists even if the hashed Diffie-Hellman KA includes the entire transcript in the final hash, i.e., the key is $H(g^a, g^b, g^{ab})$ rather than $H(g^{ab})$. The adversary would make $\Theta(q)$ $H(g^a, g^r, \star)$ queries including $h_1 := H(g^a, g^r, g^{ar})$, and after it outputs $X = g^r$ and receives $h_2$, it can still check if $h_1 = h_2$. To simulate the experiment, the reduction again has to make a guess on which $H$ query contains $g^{ar}$, even though it knows both $g^a$ and $g^r$. This attack also naturally extends to the case where the PAKE session key is $H(\phi, \phi', g^{ab})$ where $\phi = \mathcal{E}(\mathsf{pw}, g^a)$ and $\phi' = \mathcal{E}(\mathsf{pw}, g^b)$.

**Remark 3.4.** *The concrete security loss under CDH is slightly lower if the entire transcript is hashed, since the adversary's best strategy is to make $q/2$ $H(g^a, g^b, \star)$ queries and $q/2$ $H(g^a, g^r, \star)$ queries, and both of the reduction's guesses choose one of $q/2$ queries at random — incurring a $q^2/4$ loss.*[13] *By contrast, if the key is $H(g^{ab})$, then the reduction needs to guess over $q$ queries on which one would be $H(g^{ab})$ (say it is the $i$-th query) and then guess over $i$ possibilities: which of the first $i-1$ queries would be $H(g^{ar})$, or none of them would be $H(g^{ar})$ — so the loss is $\sum_{i=1}^{q} i = q(q+1)/2$. (Note that at the $i$-th query the reduction can output the query input and stop simulating the experiment, so if the $H(g^{ar})$ query happens after that, it does not matter which exact query is $H(g^{ar})$. However, the reduction does need to guess if the $H(g^{ar})$ query would happen before or after the $H(g^{ab})$ query.) Still, the loss is $\Theta(q^2)$ in both cases.*

**Remark 3.5.** *[BFGJ17] presents a comprehensive study of a large number of variants of the ODH assumption, including 1-query ODH (called sn-PRF-ODH therein*[14]*). However, their main result about 1-query ODH is in the standard model, where the RO $H(K)$ is replaced by a PRF $\mathsf{PRF}_K(x)$ (for some adversarially chosen $x$). This assumption has also been used in the security analysis of TLS [JKSS12].*

[12] The game-based proof in [BPR00] uses a variant of the CDH assumption where the adversary on $g^a, g^b$ outputs a list of group elements, and wins if one of them is $g^{ab}$. Since hashed Diffie-Hellman is tightly secure under this assumption, their proof also only has a $\Theta(q)$ security loss.

[13] Here we assume $q$ is even; if $q$ is odd, the loss is $[(q+1)/2] \cdot [(q-1)/2] = (q^2-1)/4$.

[14] [BFGJ17] considers all cases where the adversary may or may not have access to the "left oracle" $H_a(\cdot)$ that on $X$ computes $X^a$ and the "right oracle" $H_b(\cdot)$ that on $X$ computes $X^b$. In 1-query ODH there is a single query to $H_a(\cdot)$ and no query to $H_b(\cdot)$, so it is called "sn".

**Flaw in [BCP+23].** We now show the flaw in the proof of [BCP+23, Theorem 1]. This is subtle so let us proceed slowly. [BCP+23] presents a *general* statement using any 2-round KA protocol that is secure and pseudorandom (called a KEM that is indistinguishable, fuzzy, and anonymous therein), and the PAKE session key is an RO $H$ of the PAKE transcript and the KA key (see [BCP+23, Fig.5]). When instantiated with Diffie-Hellman, it becomes the "reduce to DDH" case in Table 4, so the reduction to DDH needs to make a guess over all $H$ queries. [BCP+23] uses $q_H$ to denote the number of $H$ queries[15] and $\mathsf{Adv}^{\mathrm{ind}}_{\mathrm{KEM}}(t)$ to denote the adversary's advantage against KA security; using these notations, there should be a

$$q_H \cdot \mathsf{Adv}^{\mathrm{ind}}_{\mathrm{KEM}}(t)$$

additive term in the overall distinguishing advantage of the environment. However, such a term does not appear in [BCP+23, Theorem 1]. What exactly goes wrong in the proof of this theorem?

Let us first recall the attacking scenario that causes the security loss above. The adversary passes $\phi = \mathcal{E}(\mathsf{pw}, g^a)$ from $P$ to $P'$; after that, $P'$ sends $\phi' = \mathcal{E}(\mathsf{pw}, g^b)$ aimed at $P$ and outputs its session key $H(\phi, \phi', g^{ab})$. Upon receiving $\phi'$, the adversary chooses $r \leftarrow \mathbb{Z}_p$ and computes $\phi^* = \mathcal{E}(\mathsf{pw}, g^r)$ (here the adversary needs to know $\mathsf{pw}$), gets $g^a$ by decrypting $\phi$ and makes $\Theta(q)$ $H(\phi, \phi^*, \star)$ queries including $H(\phi, \phi^*, (g^a)^r)$, and sends $\phi^*$ to $P$. After that, $P$ outputs $H(\phi, \phi^*, g^{ar})$ and the environment can check if it matches the $H$ query, and aborts if not. (We stress that the primary goal of this attack is to cause the reduction to fail, rather than to actually distinguish between the real world and the ideal world.)

The relevant hybrid in the proof is game $\mathbf{G}_{6.1}$ on pp.29–30, which says

> *On Bob's side: Upon receiving* $\mathbf{Epk}$ *from an honest Alice, instead of setting* $SK \leftarrow H(\mathsf{ssid}, P_i, P_j, \mathbf{Epk}, \mathbf{Ec}, K)$*, if* $\mathtt{SamePwd}(\mathsf{ssid}, P_i, P_j) = \mathsf{true}$*, one sets* $K' \leftarrow H^*_K(\mathsf{ssid}, \mathsf{success})$ *[...] and updates the definition* $SK \leftarrow H(\mathsf{ssid}, P_i, P_j, \mathbf{Epk}, \mathbf{Ec}, K')$.

In our terms (using the specific Diffie-Hellman KA), this means

> *On the side of* $P'$*, upon receiving* $\phi$ *from* $P$ *(passed by the adversary without modification), instead of setting the key of* $P'$ *as* $H(\phi, \phi', g^{ab})$*, if the passwords of* $P$ *and* $P'$ *are equal, one sets the key of* $P'$ *as* $H(\phi, \phi', K')$ *where* $K' \leftarrow \mathbb{G}$.

(Note that up to game $\mathbf{G}_{6.1}$, there is no change on the side of $P$ in our attacking scenario; in particular, the paragraph "On Alice's side" below specifies that "one keeps" the key of $P$.) The subsequent analysis says "we can simply successively replace $[(g^a, g^b, g^{ab})]$ with $[(g^a, g^b, K')]$, using the indistinguishability of the KEM [the DDH assumption]: the gap is bounded by $q'_{D_1} \cdot \mathsf{Adv}^{\mathrm{ind}}_{\mathrm{KEM}}(t)$." (The text in brackets is a translation to our terms.)

While this appears to be a straightforward reduction to DDH, the problem is that the reduction, given $(g^a, g^b, g^{ab})$ or $(g^a, g^b, K')$, can embed $g^{ab}$ or $K'$ while computing the session key of $P'$, but it *must simulate the rest of the experiment — that is, after* $P'$ *outputs — which involves* $g^a$. The rest of the experiment includes the following: the adversary chooses $r \leftarrow \mathbb{Z}_p$ and computes $\phi^* = \mathcal{E}(\mathsf{pw}, g^r)$, makes $\Theta(q)$ $H(\phi, \phi^*, \star)$ queries including $H(\phi, \phi^*, g^{ar})$, and sends $\phi^*$ to $P$. After that, the environment can check if the output of $P$ is $H(\phi, \phi^*, g^{ar})$. *While the simulator knows* $a$*, the reduction does not (since* $g^a$ *is part of its DDH challenge), so* $g^{ar}$ *looks random to the reduction even though it knows both* $g^a$ *and* $g^r$*. Hence, in order for the output of* $P$ *to be* $H(\phi, \phi^*, g^{ar})$*, the reduction must guess over all* $H$ *queries and lose a factor of* $\Theta(q)$. This subtle point is overlooked in [BCP+23], which misses this $\Theta(q)$ factor.

[15] In fact [BCP+23] uses the notation $q_H$ without defining it, but from the context it is clear that $q_H$ is the number of $H$ queries.

**Flaw in [DHP+18].** A very similar (yet more hidden) issue appears in the proof of [DHP+18, Theorem 6], which shows the UC-security of EKE using IC and hashed Diffie-Hellman (like [BCP+23], the entire PAKE transcript is hashed while deriving the session keys) under CDH. The problematic hybrid is game $\mathbf{G}_9$ on p.51, which says

> *$\mathcal{F}$ now generates a random session key upon a first* NewKey *query for an honest party $P_i$ with fresh record $(P_i, \mathsf{pw}_i)$ where the other party is also honest, if (at least) one of the following events happens: [...] No output was sent to the other party yet.*

In our terms, this means

> *In the case that the adversary passes the $P$-to-$P'$ message without modification, $P'$ now outputs a random session key.*

(Note that up to game $\mathbf{G}_9$, there is no change on the side of $P$ in our attacking scenario; in particular, game $\mathbf{G}_5$ deals with the case that the adversary makes an *incorrect* password guess while sending a message to $P$, and game $\mathbf{G}_6$ deals with the case that the adversary modifies the $P$-to-$P'$ message.) [DHP+18] then claims the indistinguishability between game $\mathbf{G}_9$ and the previous game in Lemma 13, whose proof is only sketched and says "it is similar to the proof of Lemma 12 [under game $\mathbf{G}_5$]" and that the reduction should simply embed $g^a, g^b$ as its CDH challenge.

However, the proofs of Lemma 13 and Lemma 12 are actually quite different. Game $\mathbf{G}_5$ says that if the adversary modifies the $P'$-to-$P$ message $\phi' = \mathcal{E}(\mathsf{pw}, g^b)$ to another $\phi^* = \mathcal{E}(\mathsf{pw}^*, \star)$, then $P$ outputs a random session key if $\mathsf{pw}^* \neq \mathsf{pw}$. The reduction to CDH here is relatively simple: since $\mathcal{D}(\mathsf{pw}, \phi^*)$ is some $g^{b^*}$ where $b^*$ is unknown to the adversary, the reduction can use it to embed a CDH challenge. (Of course, since the session key is a hash of the KA key, the reduction needs to guess over all of the adversary's $H$ queries and lose a factor of $q_H$ — which the proof correctly identifies.) However, just as what we have seen about the flaw in [BCP+23], in game $\mathbf{G}_9$ the reduction needs to simulate the rest of the experiment even after $P'$ outputs its session key, which forces the reduction to make another guess on which $H$ query is $H(\phi, \phi^*, g^{ar})$ (where the adversary sends $\phi^* = \mathcal{E}(\mathsf{pw}, g^r)$ to $P$). This part of the reduction is missing in [DHP+18].[16]

**Necessity of non-malleability.** All issues in Sects. 3.1 to 3.3 point to a property of the underlying KA protocol that is not commonly seen in the literature but is required for the security of both EKE and OEKE-PRF: Consider a "semi-man-in-the-middle" adversary that sees both protocol messages, but can only modify the second, i.e., the $P'$-to-$P$ message. Then as long as the adversary modifies the $P'$-to-$P$ message, the output of $P'$ is pseudorandom even if the adversary additionally sees the output of $P$. We call this property *non-malleability*.

**Definition 3.6.** *A KA protocol is **pseudorandom non-malleable** if the following two distributions*

[16] [DHP+18, Theorem 6] only claims the UC-security of EKE without presenting any concrete bound, so technically speaking the theorem statement is correct.

*are indistinguishable:*

| | |
|---|---|
| $a \leftarrow \mathcal{R}$ | $a \leftarrow \mathcal{R}$ |
| $b \leftarrow \mathcal{R}$ | |
| $A := \mathsf{msg}_1(a)$ | $A := \mathsf{msg}_1(a)$ |
| $B := \mathsf{msg}_2(b, A)$ | $B \leftarrow \mathcal{M}_2$ |
| $K' := \mathsf{key}_2(b, A)$ | $K' \leftarrow \mathcal{K}$ |
| $B^* \leftarrow \mathcal{A}(A, B, K')$ | $B^* \leftarrow \mathcal{A}(A, B, K')$ |
| abort if $B^* = B$ | abort if $B^* = B$ |
| $K := \mathsf{key}_1(a, B^*)$ | $K := \mathsf{key}_1(a, B^*)$ |
| output $K$ to $\mathcal{A}$ | output $K$ to $\mathcal{A}$ |

**Remark 3.7.** *The careful reader might have noticed that in Def. 3.6, the experiment on the right also changes $B$ to random — which is why we call this property* pseudorandom *non-malleability. We need this version of non-malleability for the following reason. Suppose the adversary passes the $P$-to-$P'$ message $\phi$ without modification, and on $\phi'$ from $P'$ queries $B^* := \mathcal{D}(\mathsf{pw}^*, \phi')$ on some password guess $\mathsf{pw}^*$. In the real world $B^*$ is the "real" $\mathsf{msg}_2(b, A)$ if $\mathsf{pw}^* = \mathsf{pw}$, and uniformly random otherwise. However, in the ideal world* the simulator does not know whether $\mathsf{pw}^*$ is the correct password at this point*, so its simulation must be indistinguishable from both cases. By transitivity, the adversary must not distinguish $B := \mathsf{msg}_2(b, A)$ from $B \leftarrow \mathcal{M}_2$. In other words, the* joint distribution *of $B$ and $K'$ must be indistinguishable from random, even to an adversary that gets to modify $\phi'$ and see what session key $P$ then generates.*

*Of course, in the Diffie-Hellman example there is no difference between computing $B$ as $B := \mathsf{msg}_2(b, A)$ and sampling $B \leftarrow \mathcal{M}_2$, as the messages in Diffie-Hellman KA are uniform. But in the general case there is indeed a difference (even assuming the KA protocol satisfies strong pseudorandomness), as we show in Appx. B.*

**Remark 3.8.** *Recall that (standard, not strong) second pseudorandomness says that $B := \mathsf{msg}_2(b, A)$ and $B \leftarrow \mathcal{M}_2$ are indistinguishable, even given $A := \mathsf{msg}_1(a)$. This is of course implied by pseudorandom non-malleability. We present second pseudorandomness as a separate property for clarity, and also because it is a standard property that was mentioned in several prior works (while pseudorandom non-malleability was not).*

### 3.4 Allowing Identity Element in Diffie-Hellman Makes OEKE Insecure

There is a separate (and simpler) attack on OEKE-PRF if the underlying KA protocol is plain Diffie-Hellman: the adversary (that does not know the password) disregards the $P$-to-$P'$ message and sends $(e, \mathsf{PRF}_e(1))$ to $P$, where $e$ is the identity group element. Then $P$ computes $K = e^a = e$, so the check passes and $P$ outputs session key $\mathsf{PRF}_e(0)$ — which the adversary can predict. This attack can be easily generalized to any version of OEKE using plain Diffie-Hellman, and in particular OEKE-RO is also insecure; and it extends to OEKE using hashed Diffie-Hellman (the adversary sends $(e, \mathsf{PRF}_{H(e)}(1))$ to $P$). An obvious fix is to disallow $e$ as a KA message, i.e., sample the exponents $a, b$ from $\mathbb{Z}_p \setminus \{0\}$ rather than $\mathbb{Z}_p$.

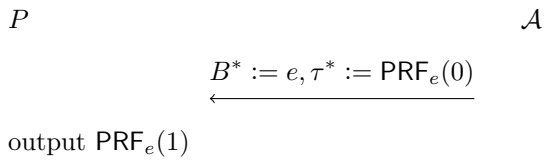


Figure 8: Attack on OEKE-PRF with plain Diffie-Hellman. $\mathcal{A}$ (that does not know $\mathsf{pw}$) sends a single message to $P$ and can predict the session key of $P$

This attack shows that the underlying KA protocol must satisfy some form of *contributoryness* property, namely both parties must "contribute" to the output key and a single party cannot bias the distribution of the output key too much. We formalize the necessary property which we call *collision resistance*. [SGJ23, Theorem 3] only requires the KA protocol to be secure and pseudorandom; in other words, it allows for using plain Diffie-Hellman — where $e$ can be a message — in OEKE, so this theorem is false. (As we have seen in Sect. 1.1, it is not even clear what [SGJ23, Theorem 3] exactly is, since the theorem statement is about OEKE-RO while its proof is about OEKE-PRF. But our attack shows that [SGJ23, Theorem 3] is false no matter what it means.) Note that the above attack does not apply to EKE where the $P'$-to-$P$ message $g^b$ is encrypted under $\mathsf{pw}$ (so sending $\mathcal{E}(\mathsf{pw}, e)$ requires knowledge of $\mathsf{pw}$).

**Definition 3.9.** *A KA protocol is* ***collision-resistant*** *if the key space is $\mathcal{K} = \{0,1\}^{3\kappa}$, and for any polynomially bounded $q$ and any PPT adversary $\mathcal{A}$, the winning probability of $\mathcal{A}$ in the following game is negligible:*

$$\boxed{\begin{array}{l}\forall i \in [q]\colon a_i \leftarrow \mathcal{R} \\ \forall i \in [q]\colon A_i := \mathsf{msg}_1(a_i) \\ B^* \leftarrow \mathcal{A}(A_1, \ldots, A_q) \\ \forall i \in [q]\colon K_i \parallel \tau_i := \mathsf{key}_1(a_i, B^*) \\ \mathcal{A} \text{ wins if } \exists_{i \neq j}\, \tau_i = \tau_j\end{array}}$$

*Here, we split keys $\mathsf{key}_1(a_i, B^*) \in \{0,1\}^{3\kappa}$ into two chunks: $K \in \{0,1\}^{\kappa}$ and $\tau \in \{0,1\}^{2\kappa}$.*[17]

**Remark 3.10.** *In OEKE-RO, if the authenticator $\tau$ is defined as $H(\mathsf{pw}', K', 1)$ rather than $H(K', 1)$, then collision resistance is unnecessary since the adversary needs to know the correct password in order to generate a valid authenticator, and the simulator can extract the password from the adversary's $H$ queries. The OEKE-RO protocol analyzed in [BCP+23, Theorem 2] uses this approach, so it does not suffer from the problem in this section.*

### 3.5 EKE Using HIC/POPF Only Realizes a Weaker UC Functionality

All attacks we have discussed work for (O)EKE no matter whether the underlying encryption scheme is IC, HIC or POPF; in this section we present an attack that applies to EKE using HIC and POPF only. Since we have not formally introduced the concepts of HIC and POPF, we are satisfied with giving an informal argument, and do not prove the UC-insecurity (like what we did in Thm. 3.1).

A key difference between HIC/POPF and IC is that the encryption algorithm in the latter is deterministic, whereas the former is *randomized*; that is, in HIC/POPF the output of $\mathcal{E}(\mathsf{pw}, m; r)$

[17] We require $\tau$ to be $2\kappa$-bit long, as an adversary can win the experiment with probability roughly $q^2/2^{|\tau|}$. Technically this renders OEKE-PRF impossible, as $K$ and $\tau$ now have different lengths. However, we can consider a version of OEKE-PRF where $K = \mathsf{PRF}_{\bar{K}}(0)$ and $\tau = \mathsf{PRF}_{\bar{K}}(1) \parallel \mathsf{PRF}_{\bar{K}}(2)$ (where $\bar{K}$ is the $\kappa$-bit key of the underlying KA protocol). Alternatively, we can stick to a $\kappa$-bit $\tau$ if we allow the adversary's advantage to be quadratic in $\kappa$.

depends on the randomness $r$ used in the algorithm. This yields the following attack: the adversary passes the $P$-to-$P'$ message without modification, causing $P'$ to output session key $K'$. Then on the $P'$-to-$P$ message $\phi' = \mathcal{E}(\mathsf{pw}, B; r')$, the adversary sets $\mathsf{pw}^* = \mathsf{pw}$ with probability $1/2$ and $\mathsf{pw}^* \neq \mathsf{pw}$ with probability $1/2$, and decrypts and re-encrypts using $\mathsf{pw}^*$. That is, the adversary computes $B^* = \mathcal{D}(\mathsf{pw}^*, \phi')$ and sends

$$\phi^* := \mathcal{E}(\mathsf{pw}^*, B^*; r^*)$$

to $P$, where $r^*$ is a fresh randomness sampled from the corresponding space. Let $K$ be the session key of $P$; $K$ is equal to $K'$ with probability $1/2$ (if $\mathsf{pw}^* = \mathsf{pw}$) and independent of $K'$ with probability $1/2$ (if $\mathsf{pw}^* \neq \mathsf{pw}$).

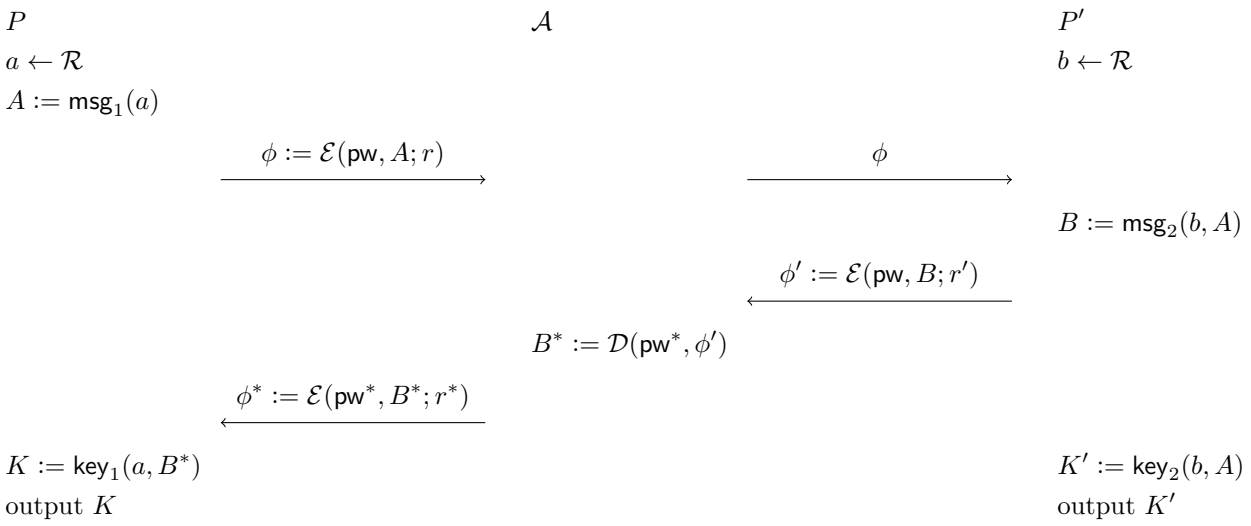


Figure 9: Attack on EKE using HIC/POPF (with any KA protocol). $\mathcal{A}$ sets $\mathsf{pw}^* = \mathsf{pw}$ with probability $1/2$ and $\mathsf{pw}^* \neq \mathsf{pw}$ with probability $1/2$. Either way $\phi^* \neq \phi'$ with overwhelming probability due to the fresh $r^*$, but in the former case $K = K'$ and in the latter case $K$ and $K'$ are independent. This "second round conditional password test" is not allowed by standard UC-security

The simulator can extract $\mathsf{pw}^*$ in the second round (which happens after $P'$ outputs $K'$), but it does not know whether $\mathsf{pw}^* = \mathsf{pw}$ unless and until it sends $(\mathsf{TestPwd}, \mathsf{sid}, P, \mathsf{pw}^*)$ to $\mathcal{F}_{\mathrm{PAKE}}$. The simulator has two options here. If it sends the TestPwd command, then the simulation fails in the case of $\mathsf{pw}^* = \mathsf{pw}$, since (as we argued in Sect. 3.1) $K'$ is independent of the simulator's view, so after compromising the $P$ session, it cannot set $K$ to be equal to $K'$. If the simulator does not send the TestPwd command, then the simulation fails in the case of $\mathsf{pw}^* \neq \mathsf{pw}$, as the $P$ session is fresh and $\mathcal{F}_{\mathrm{PAKE}}$ will let $P$ output the same session key as $P'$, while the two session keys are independent in the real world. Either way, the simulator fails with probability roughly $1/2$. (Note that this is not an issue if we use IC, since the simulator can detect whether $\mathsf{pw}^* = \mathsf{pw}$ by observing whether $\phi^* = \phi'$, and send TestPwd only if $\mathsf{pw}^* \neq \mathsf{pw}$. Here the randomization of HIC/POPF ensures that even if $\mathsf{pw}^* = \mathsf{pw}$, $\phi^*$ and $\phi$ are still independent.)

The above shows that [MRR20, Theorem 10] and [SGJ23, Theorem 2] are false in a manner different from Sect. 3.1, since these two theorems imply that EKE with plain Diffie-Hellman is secure using POPF and HIC, respectively. (In fact the same attack works even if we use hashed Diffie-Hellman, if only the key is hashed.) Closer scrutiny shows that EKE using HIC/POPF realizes a weaker UC functionality, with an additional command TestSamePwd that works in the same way

as TestPwd except that no change is made if $\mathsf{pw}^* = \mathsf{pw} = \mathsf{pw}'$ (where $\mathsf{pw}'$ is the key of $P'$). (In the attacking scenario above, this allows the simulation to go through in the case of $\mathsf{pw}^* = \mathsf{pw}$, as the TestSamePwd command does not have any effect and the $P$ session is still fresh, so $\mathcal{F}_{\text{PAKE}}$ will set $K$ to be equal to $K'$.) We call this modified PAKE functionality *PAKE with same password test*, or $\mathcal{F}_{\text{PAKE-sp}}$.

Of course, another way to get around this issue is to modify the protocol so that the $P'$-to-$P$ message is included in the final hash, or rather (if we do not want to explicitly use the ROM here) to define the session key as $\mathsf{PRF}_K(\phi')$. We call this modified protocol EKE-PRF. Furthermore, as we noticed above, the attack does not work if IC is used. In this work we present three results:

1. EKE (with the underlying KA satisfying appropriate properties, same below) using IC realizes $\mathcal{F}_{\text{PAKE}}$;
2. EKE using POPF realizes $\mathcal{F}_{\text{PAKE-sp}}$;
3. EKE-PRF using POPF realizes $\mathcal{F}_{\text{PAKE}}$.

Our main result is (3), and we briefly argue (1) and (2) in Appx. D. For (2), we believe our $\mathcal{F}_{\text{PAKE-sp}}$ functionality might be of independent interest; for (1), we believe it would be helpful to present a (correct) proof for EKE using IC, as this is the most studied version of EKE so far.

**History of the attack.** Like the attack in Sect. 3.1, the attack in this section was first mentioned in Jarecki's EUROCRYPT talk [Jar23] and is not our original work. The modification of the protocol (item (3) above) was also suggested in the talk. The modification of the UC PAKE functionality (item (2) above) is ours.

### 3.6 Summary

We summarize the requirements on the underlying KA protocol for (O)EKE in Table 5. (For a discussion of strong pseudorandomness, see Appx. A.)

| | security and pseudorandomness | strong pseudorandomness | pseudorandom non-malleability | collision-resistance |
|---|---|---|---|---|
| EKE-PRF (or EKE if IC is used) | ✓ | ✓ (overlooked in [MRR20, BCP+23]) | ✓ (overlooked in [MRR20, SGJ23] and security analyses in [DHP+18, BCP+23]) | |
| OEKE | ✓ | | ✓ (overlooked in [SGJ23, BCP+23]) | ✓ (overlooked in [SGJ23]) |
| EKE using HIC/POPF does not realize $\mathcal{F}_{\text{PAKE}}$ no matter what KA protocol is used, which is overlooked in [MRR20, SGJ23] | | | | |

Table 5: Requirements on the underlying KA protocol in (O)EKE

We remark that the UC-security of EKE has long been a "folklore" result in the community, before a formal proof was presented. However, it seems unclear which exact version of EKE was understood to be UC-secure, and as [MRR20, Theorem 10] and [SGJ23, Theorem 2] suggest, some

might have held the false belief that EKE with plain Diffie-Hellman is UC-secure (while others seem to have the correct understanding that the Diffie-Hellman output has to be hashed). This reveals the problem with such "folklore" results: without a formal analysis, people cannot even agree upon what the result exactly is!

**Beyond Diffie-Hellman.** All attacks in this section assume the (O)EKE protocol uses some variants of the Diffie-Hellman KA. This significantly simplifies our presentation, since Diffie-Hellman has perfect pseudorandomness (i.e., the protocol messages are uniform). But this also means that in this section we are not able to show some additional attacks when a general KA protocol is used. These additional subtleties come from two dimensions: (1) As we have seen in Sect. 2.1 and explain in Appx. A, the security of EKE requires strong pseudorandomness, which is not implied by (standard) pseudorandomness in the general case; and (2) as we have seen in Sect. 3.3 and explain in Appx. B, the security of both EKE and OEKE requires pseudorandom non-malleability, which is not implied by security and strong pseudorandomness in the general case, even if we apply an RO to derive the key and allow for non-tight reductions. These issues are in no way less intriguing and perhaps even more important, as post-quantum KAs such as Kyber do not have perfect pseudorandomness. However, we feel this section is already too long. For the additional subtleties and attacks in the general case, see Appxs. A and B.

# 4 Almost Universally Composable POPF

As mentioned in Sect. 1.2, the POPF definition in [MRR20] is unusual in requiring a higher-order security definition, saying that the POPF must remain secure when used with any weak PRF. We use a similar idea, but generalize it significantly, providing something that is almost a UC definition for POPFs. We give an ideal functionality $\mathcal{F}_{\text{POPF}}$, and define a POPF to be a protocol that can be composed with any protocol in the $\mathcal{F}_{\text{POPF}}$-hybrid model that satisfies a certain condition.

More formally:

**Definition 4.1.** *Let $\mathcal{F}_0, \mathcal{F}_1$ be two ideal functionalities, which must make no reference to a global RO $R$. Let $\pi$ be a protocol instantiating $\mathcal{F}_1$ in the $(\mathcal{F}_{\text{POPF}}, \mathcal{F}_0)$-hybrid world. A protocol $\pi_{\text{POPF}}$ is a POPF if the composed protocol $\pi' = \pi \diamond \pi_{\text{POPF}}$ realizes $\mathcal{F}_1$ in the $\mathcal{F}_0$-hybrid world, for all such $\mathcal{F}_0, \mathcal{F}_1, \pi$.*

## 4.1 The Functionality $\mathcal{F}_{\text{POPF}}$

A POPF can be thought of as a family of random functions $\{F_\phi\}$, with two interfaces: Program, where a party picks a function $F_\phi$ with $F_\phi(x^*) := y^*$ for $(x^*, y^*)$ of its choice, and Eval, where a party evaluates a function $F_\phi$ on a specific input $x$. Crucially, every honest function $F_\phi$ can be programmed at only one point $(x^*, y^*)$, and for any $x \neq x^*$, $F_\phi$ is random. Furthermore, POPFs must satisfy the *uncontrollable output* property: for an adversarially generated $\phi^*$, the output of $F_{\phi^*}(x)$ is pseudorandom — in particular, it is a suitable input of some higher-level schemes which use random inputs (e.g., a weak PRF, as in the definition in [MRR20]) — except on a single $x$ "extracted" during the evaluation of $F_{\phi^*}(x)$.

Our POPF ideal functionality, $\mathcal{F}_{\text{POPF}}$, is shown in Figure 10. $\mathcal{F}_{\text{POPF}}$ maintains a set $T$ of defined function values, where $(\phi, x, y) \in T$ means that $F_\phi(x) = y$. When party $P$ (either honest or malicious) wants to program on a pair $(x^*, y^*)$, $\mathcal{F}_{\text{POPF}}$ lets the adversary specify a function index $\phi$. In particular, if there are no functions $F_\phi$ in the family such that $F_\phi(x^*)$ is set to be $y^*$, the adversary must choose a *new* index $\phi$; this ensures that $F_\phi$ is never programmed on two

Parameters:

- Random oracle $R : \{0,1\}^\kappa \times \mathcal{X} \to \mathcal{Y}$ (modeled by relativizing the UC framework with an RO that all interactive Turing machines may access).

Global variables (per $\mathsf{sid}$):

- Transcript $T = \{\}$ of POPF evaluations.
- Set $\Phi = \{\}$ of honest POPF indices.
- Malicious POPF $\phi^* = \bot$.
- A string $\alpha^* = \bot$ representing a key for $R$.

On $(\mathsf{Program}, \mathsf{sid}, x^*, y^*)$ from party $P$ (where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$):

1. There are two cases, depending on whether there is an entry $(\cdot, x^*, y^*) \in T$.
   - A. If there is no such entry, or if $P$ is malicious, send $(\mathsf{Program}, \mathsf{sid})$ to $\mathcal{A}^*$. Wait until $\mathcal{A}^*$ responds with $(\mathsf{Program}, \mathsf{sid}, \phi)$ such that there is no entry $(\phi, \cdot, \cdot) \in T$. Then add $\phi$ to $\Phi$.
   - B. Otherwise, there is such an entry, and $P$ is honest. Send $(\mathsf{Program}, \mathsf{sid}, \{\phi \mid (\phi, x^*, y^*) \in T\})$ to $\mathcal{A}^*$. Wait until $\mathcal{A}^*$ responds with $(\mathsf{Program}, \mathsf{sid}, \phi)$ such that $(\phi, \cdot, \cdot) \notin T$ or $(\phi, x^*, y^*) \in T$.
2. If $(\phi, x^*, y^*) \notin T$, add $(\phi, x^*, y^*)$ to $T$.
3. Send $(\mathsf{Program}, \mathsf{sid}, \phi)$ to $P$.

On $(\mathsf{Eval}, \mathsf{sid}, \phi, x)$ from party $P$ (where $x \in \mathcal{X}$):

4. If $\phi \notin \Phi$, $\phi^* = \bot$, and $P$ is honest, then:
   - (1) Send $(\mathsf{Extract}, \mathsf{sid}, \phi)$ to $\mathcal{A}^*$ and wait for response $(\mathsf{Extract}, \mathsf{sid}, x^*, \alpha)$.
   - (2) Send $(\mathsf{Eval}, \mathsf{sid}, \phi, x^*)$ to $\mathcal{A}^*$ and wait for response $(\mathsf{Eval}, \mathsf{sid}, y^*)$.
   - (3) Set $\phi^* := \phi$, $\alpha^* := \alpha$, and add $(\phi, x^*, y^*)$ to $T$.
5. Check if there is an entry $(\phi, x, y) \in T$. If not, generate $y$ according to three cases:
   - A. If $\phi \in \Phi$, sample $y \leftarrow \mathcal{Y}$.
   - B. If $\phi = \phi^*$, let $y := R(\alpha^*, x)$.
   - C. Otherwise, send $(\mathsf{Eval}, \mathsf{sid}, \phi, x)$ to $\mathcal{A}^*$ and wait for response $(\mathsf{Eval}, \mathsf{sid}, y)$.

   Finally, add $(\phi, x, y)$ to $T$.
6. Send $(\mathsf{Eval}, \mathsf{sid}, y)$ to $P$.

Figure 10: Ideal functionality $\mathcal{F}_{\text{POPF}}$ (with domain $\mathcal{X}$ and range $\mathcal{Y}$).

distinct points. $\mathcal{F}_{\text{POPF}}$ also adds $\phi$ to the set of honest POPF indices $\Phi$, which indicates that $F_\phi$ is "programmable-once" and has been programmed on one input/output pair. On the other hand, if there are such functions $F_\phi$ such that $F_\phi(x^*) = y^*$ (and $P$ is honest), $\mathcal{F}_{\text{POPF}}$ lets the adversary know all such indices $\phi$.[18] Then the adversary has two options: either pick one of these existing indices (i.e., the programming process is identical to a previous one), or choose a *new* index $\phi$, just as in the previous case.

When a party $P$ wants to evaluate $F_\phi(x)$, whose result has not been defined through $T$, $\mathcal{F}_{\text{POPF}}$ has several cases based on how $\phi$ was generated.

- If $\phi$ is an honest index, this means that $F_\phi$ is "programmable-once" and has already been programmed on another input/output pair, so it chooses $F_\phi(x)$ at random.
- If $\phi$ is the "designated malicious index" $\phi^*$, we want to use $F_\phi(x)$ as the input of some higher-level scheme. The adversary should be able to learn $F_\phi(x)$, but not control it when $x$ differs from its chosen target $x^*$. Therefore, we set $F_\phi(x)$ to be the output of a *global* RO $R$, queried on $\phi$ and $x$. (Note that this is weaker than choosing the output at random.)
- Otherwise, i.e., if $\phi$ is a malicious index other than $\phi^*$, $\mathcal{F}_{\text{POPF}}$ allows the adversary to program $F_\phi$ on all inputs; in particular, $\mathcal{F}_{\text{POPF}}$ asks the adversary for $F_\phi(x)$.

The "designated malicious index" $\phi^*$ is chosen as the first malicious POPF index $\phi^*$ given to $\mathsf{Eval}$ by an honest party. Having only a single designated target is a necessary limitation of our POPF construction — requiring that multiple POPFs have jointly uncontrollable outputs is much more stringent than just requiring a single POPF to be uncontrollable. In all existing protocols that use POPFs, they only need to be individually uncontrollable and not jointly uncontrollable, so this limitation is not too onerous.

**On the global random oracle $R$.** Step 5B deserves further explanation. It sets $F_{\phi^*}(x) = R(\alpha^*, x)$ for some key $\alpha^*$ and a global RO $R$. Why is there a key $\alpha^*$? Without it, in the real world the environment could query $R(x)$, and check that it matches $F_{\phi^*}(x)$. Neither $R$ nor $F_{\phi^*}(x)$ can be programmed to match the other in the real world, so it would notice the inconsistency. $\alpha^*$ solves this, as if the POPF's simulator samples it randomly, the environment cannot query $R(\alpha^*, x)$ except with negligible probability.

But this brings up a different question: why use a global RO at all? After all, $\alpha^*$ would be unnecessary if $\mathcal{F}_{\text{POPF}}$ just sampled $F_{\phi^*}(x)$ randomly, like in a local RO. But using a local RO would allow the simulator $\mathcal{S}$ of any higher-level protocol $\pi$ to program the POPF outputs, so the POPF simulator would have to be able to program $F_{\phi^*}(x)$ for *any* maliciously chosen $\phi^*$. A global RO, on the other hand, cannot be controlled by the higher-level scheme. Instantiating $\mathcal{F}_{\text{POPF}}$ in the standard UC framework would still require programming $F_{\phi^*}(x)$, but to satisfy Def. 4.1 we only need to program it inside a hybrid. The global RO (or at least $R(\alpha^*, \cdot)$) can later be replaced with $F_{\phi^*}(x)$, because it can only be queried by $\mathcal{S}$; Def. 4.1 requires that the higher-level functionality $\mathcal{F}_1$ not reference $R$. This means that the malicious POPF index only needs to be programmed inside a hybrid, and not in the final simulator, so this programming can use techniques not allowed in UC.

[18] A malicious party could exploit this to find whether a given point $(x^*, y^*)$ has been evaluated, so we only allow honest parties to trigger this case.

### 4.2 POPF Construction

Our POPF construction is identical to the 2-round Feistel POPF in [MRR20]. It uses an abelian group $\mathbb{G}$[19] and two ROs $H : \{0,1\}^* \times \{0,1\}^{3\kappa} \to \mathbb{G}$ and $H' : \{0,1\}^* \times \mathbb{G} \to \{0,1\}^{3\kappa}$. The function family is indexed by $\phi = (s,t) \in \{0,1\}^{3\kappa} \times \mathbb{G}$ and defined as

$$F_\phi(x) = H(x, s \oplus H'(x,t)) \cdot t.$$

To program on an input/output pair $(x^*, y^*)$, one can compute $\phi = (s,t)$ via the following process: sample $r \leftarrow \{0,1\}^{3\kappa}$, and solve the equations

$$\begin{cases} H(x^*, r) \cdot t = y^*, \\ s \oplus H'(x^*, t) = r \end{cases}$$

for first $t$, then $s$. Note that for a pair of $(x^*, y^*)$, there are exponentially many $\phi$ such that $F_\phi(x^*) = y^*$. This is a crucial difference from the ideal cipher, where the encryption is deterministic.

The formal description of our construction is shown in Figure 11. Both of the two ROs are implemented in the same ideal functionality $\mathcal{F}_{\text{RO}}$. To simplify the notation, we have left the session id $\mathsf{sid}$ implied in queries to $H$ and $H'$.

Parameters:
- Random oracle functionality $\mathcal{F}_{\text{RO}}$ (see Figure 12).

On $(\mathsf{Program}, \mathsf{sid}, x^*, y^*)$ (where $x^* \in \mathcal{X}$ and $y^* \in \mathbb{G}$):
1. Choose $r \leftarrow \{0,1\}^{3\kappa}$.
2. Query $h := H(x^*, r)$, and compute $t := y^*/h$.
3. Query $h' := H'(x^*, t)$, and compute $s := r \oplus h'$.
4. Output $(\mathsf{Program}, \mathsf{sid}, (s,t))$.

On $(\mathsf{Eval}, \mathsf{sid}, (s,t), x)$ (where $x \in \mathcal{X}$, $s \in \{0,1\}^{3\kappa}$, and $t \in \mathbb{G}$):
5. Query $h' := H'(x, t)$, and compute $r := s \oplus h'$.
6. Query $h := H(x, r)$, and compute $y := h \cdot t$.
7. Output $(\mathsf{Eval}, \mathsf{sid}, y)$.

Figure 11: POPF construction $\pi_{\text{POPF}}$ (with domain $\mathcal{X}$ and range $\mathbb{G}$). The ROs $H, H'$ are queried using $\mathcal{F}_{\text{RO}}$

### 4.3 Security Analysis

**Theorem 4.2.** *The protocol $\pi_{\text{POPF}}$ is a POPF. That is, for any protocol $\pi$ instantiating $\mathcal{F}_1$ in the $(\mathcal{F}_{\text{POPF}}, \mathcal{F}_0)$-hybrid world with distinguisher advantage $\mathsf{Adv}_\pi$, the composed protocol $\pi \diamond \pi_{\text{POPF}}$ realizes $\mathcal{F}_1$ in the $\mathcal{F}_0$-hybrid world.*

#### 4.3.1 The Pseudo-simulator

As the first step of our security analysis, we describe in Figure 13 what we call the "pseudo-simulator" for $\pi_{\text{POPF}}$, $\mathcal{S}_{\text{pseudo}}$. $\mathcal{S}_{\text{pseudo}}$ can be viewed as an attempt to prove that $\pi_{\text{POPF}}$ realizes $\mathcal{F}_{\text{POPF}}$;

[19] In this section the group $\mathbb{G}$ does not need to be cyclic, which is different from the Diffie-Hellman group in Sects. 2 and 3. In particular, the EKE and OEKE protocols in Sect. 5 can work in a non-cyclic Abelian group when instantiated with the POPF in this section. To make this distinction clear, in this section we simply use $|\mathbb{G}|$ (rather than $p$) for the group order.

Parameters:

- Abelian group $\mathbb{G}$ with order $2^{2\kappa} \leq |\mathbb{G}| < 2^{2\kappa+1}$.

Global variables:

- Initialize a *list* $T_{\text{RO}} := []$ of RO evaluations.

On $H(x, r)$ (for session sid) from party $P$:

1. Ignore this query if $r \notin \{0,1\}^{3\kappa}$.
2. If there is not a matching query "$h = H(x, r)$" $\in T_{\text{RO}}$, sample $h \leftarrow \mathbb{G}$ and append "$h = H(x, r)$" to $T_{\text{RO}}$.
3. Return $h$ to $P$.

On $H'(x, t)$ (for session sid) from party $P$:

1. Ignore this query if $t \notin \mathbb{G}$.
2. If there is not a matching query "$h' = H'(x, t)$" $\in T_{\text{RO}}$, sample $h' \leftarrow \{0,1\}^{3\kappa}$ and append "$h' = H'(x, t)$" to $T_{\text{RO}}$.
3. Return $h'$ to $P$.

Figure 12: Ideal functionality $\mathcal{F}_{\text{RO}}$.

however, to successfully program a maliciously generated POPF's output to match the RO $R$, $\mathcal{S}_{\text{pseudo}}$ has to make a random guess over all of the adversary's $H'$ queries, and its simulation is successful only if the guess is correct. As such, $\mathcal{S}_{\text{pseudo}}$ is not a valid simulator showing that $\pi_{\text{POPF}}$ realizes $\mathcal{F}_{\text{POPF}}$, but it is a critical step towards building the actual simulator presented next.

Below we explain how $\mathcal{S}_{\text{pseudo}}$ works. It can be divided into two separate goals: simulating honestly generated POPFs without knowing where they were programmed, and forcing maliciously generated POPFs to have output matching the global RO $R$.

**Honest POPFs.** When $\mathcal{S}_{\text{pseudo}}$ is asked to program on an input/output pair (which $\mathcal{S}_{\text{pseudo}}$ does not know), it simply chooses a random $\phi = (s, t)$. It then needs to answer the adversary's RO queries appropriately, so that evaluating the POPF via the RO gives the same result as calling Eval on the ideal functionality. On an $H(x, r)$ query, $\mathcal{S}_{\text{pseudo}}$ checks if $r = s \oplus H'(x, t)$ for some *honest* $\phi = (s, t)$, i.e., the adversary is trying to compute $y = F_\phi(x) = H(x, r) \cdot t$. If so, then $\mathcal{S}_{\text{pseudo}}$ sends an Eval command to $\mathcal{F}$ to obtain $y$, and then programs $H(x, r) := y/t$.

**Malicious POPFs.** On the other hand, the $H(x, r)$ query might be the adversary computing $F_{\phi^*}(x)$ for the "designated malicious index" $\phi^* = (s^*, t^*)$. Note that $\phi^*$ might not have been chosen yet — $\mathcal{S}_{\text{pseudo}}$ only learns $\phi^*$ when Extract is called. It only knows that if this $H$ query is evaluating $F_{\phi^*}(x)$ then $H'(x^*, t^*)$ must have been queried to compute $s^*$, and $H'(x, t^*)$ must have been queried to find $r$. Therefore, $\mathcal{S}_{\text{pseudo}}$ *has to choose a guess* $t_g$ *for what* $t^*$ *will be, among all of the adversary's* $H'$ *queries.* Then $\mathcal{S}_{\text{pseudo}}$ guesses $s^*$ by solving for $s_g = H'(x, t_g) \oplus r$, obtains $y$ by explicitly querying the *global RO* $R$[20], and programs $H(x, r) := y/t_g$. In all other cases, $\mathcal{S}_{\text{pseudo}}$ answers the RO queries by lazy sampling.

Later, Extract will be called, and $\mathcal{S}_{\text{pseudo}}$ will have to find the unique point $x^*$ programmed by the POPF. Recall that to find $(s^*, t^*)$, the adversary needs to compute first $t^* = y^*/H(x^*, r)$, then $s^* = r \oplus H'(x^*, t^*)$; the query $H'(x^*, t^*)$ is called the *anchor query.* That is, $\mathcal{S}_{\text{pseudo}}$ identifies the

[20] If $\mathcal{S}_{\text{pseudo}}$ sends an Eval command to $\mathcal{F}$, $\mathcal{F}$ might enter step 5C in Fig. 10 (note that it will not enter step 4 since it is $\mathcal{S}_{\text{pseudo}}$ that sends the command, and will not enter step 5A since $\phi^*$ is a malicious index), in which $\mathcal{F}$ will send an Eval command back to $\mathcal{S}_{\text{pseudo}}$ and cause a loop.

adversary's anchor query as the query $(x^*, t^*)$ to $H'$ (resulting in $h^*$) such that $H(x^*, s^* \oplus h^*)$ was queried before the $H'$ query. However, note that there is an exception: given any $\phi = (s, t)$, the adversary can choose another index $\phi' = (s_\phi, t_\phi)$ and input $x^*$ such that

$$s_\phi \oplus H'(x^*, t_\phi) = s \oplus H'(x^*, t),$$

causing

$$F_{\phi'}(x^*) = F_\phi(x^*) \cdot (t_\phi / t)$$

without making any $H$ query. In this case, the anchor query is defined as the query $H'(x^*, t)$. Either way, $\mathcal{S}_{\text{pseudo}}$ outputs $x^*$ in the anchor query, which implicitly sets $F_\phi(x^*)$ because the ideal functionality sends an Eval command to the simulator when $F_\phi(x^*)$ is evaluated. Finally, to answer an Eval command, $\mathcal{S}_{\text{pseudo}}$ simply returns the honestly computed function value (note that this might trigger a fresh $H$ query, and how to answer it is described at the beginning of this paragraph).

The guessing step while answering $H'$ queries is why the pseudo-simulator does not (quite) work, as the probability of a correct guess is $1/(q'_h + 1)$ (including the additional guess that Extract will never be called or that $H'(x, t^*)$ wasn't queried.). Since we cannot prove that $\pi_{\text{POPF}}$ realizes $\mathcal{F}_{\text{POPF}}$, we turn to the "almost UC" notion of Def. 4.1.

### 4.3.2 The Simulator

Let $\mathcal{F}_0, \mathcal{F}_1$ be two ideal functionalities which make no reference to $R$, and $\pi$ be a protocol instantiating $\mathcal{F}_1$ in the $(\mathcal{F}_{\text{POPF}}, \mathcal{F}_0)$-hybrid model. Let $\mathcal{S}$ be the simulator for $\pi$, i.e., $\pi \diamond (\mathcal{F}_{\text{POPF}}, \mathcal{F}_0)$ and $\mathcal{F}_1 \diamond \mathcal{S}$ are indistinguishable. We construct the POPF simulator $\mathcal{S}_{\text{POPF}}$ in Figure 14.

### 4.3.3 Security Proof

**Proof overview.** To prove that $\pi_{\text{POPF}}$ is a POPF (Def. 4.1), we must show that its composition with $\pi$ is simulated by $\mathcal{S}_{\text{POPF}}$. If $\pi_{\text{POPF}}$ realized $\mathcal{F}_{\text{POPF}}$, we could simply apply the UC theorem. Unfortunately, it does not, because of the aforementioned aborts in the pseudo-simulator $\mathcal{S}_{\text{pseudo}}$.

Our proof has a similar structure to the proof of the UC theorem, but adjusted to take into consideration the aborts inside $\mathcal{S}_{\text{pseudo}}$. Recall that the UC theorem states that, if an "inner protocol" $\pi_{\text{inner}}$ realizes functionality $\mathcal{F}_{\text{inner}}$ (let $\mathcal{S}_{\text{inner}}$ be the simulator), and an "outer protocol" $\pi$ realizes functionality $\mathcal{F}_1$ in the $\mathcal{F}_{\text{inner}}$-hybrid world (let $\mathcal{S}$ be the simulator), then the combined protocol $\pi \diamond \pi_{\text{inner}}$ also realizes $\mathcal{F}_1$. At a high level, the hybrid proof goes through the following steps:

1. Real world: $\pi \diamond \pi_{\text{inner}}$.
2. Intermediate world: $\pi \diamond \mathcal{F}_{\text{inner}} \diamond \mathcal{S}_{\text{inner}}$. This is indistinguishable from the real world, since $\pi_{\text{inner}}$ realizes $\mathcal{F}_{\text{inner}}$.
3. Ideal world: $\mathcal{F}_1 \diamond \mathcal{S} \diamond \mathcal{S}_{\text{inner}}$. This is indistinguishable from the intermediate world, since $\pi$ realizes $\mathcal{F}_1$. The simulator we need is essentially a combination of $\mathcal{S}_{\text{inner}}$ and $\mathcal{S}$.

In our setting, the "inner protocol" is $\pi_{\text{POPF}}$, whose simulator $\mathcal{S}_{\text{pseudo}}$ has high probability of abort. Thus, we add aborts to our real and ideal worlds, and show that the "real world with abort" and the "ideal world with abort" are indistinguishable. Below we use the notation $p\mathcal{W}$ to represent running process $\mathcal{W}$ with probability $p$, and aborting instead with probability $1 - p$.

1. Real world with abort: $\frac{1}{q'_h+1}(\pi \diamond (\pi_{\text{POPF}}, \mathcal{F}_{\text{RO}}, \mathcal{F}_0))$.
2. Pseudo-intermediate world: $\pi \diamond (\mathcal{F}_{\text{POPF}} \diamond \mathcal{S}_{\text{pseudo}}, \mathcal{F}_0)$. Assuming that $\mathcal{S}_{\text{pseudo}}$ does not abort, $\pi_{\text{POPF}} \diamond \mathcal{F}_{\text{RO}}$ is indistinguishable from $\mathcal{F}_{\text{POPF}} \diamond \mathcal{S}_{\text{pseudo}}$, so this is indistinguishable from the

Initialize a *list* $T_{\mathrm{RO}} := []$ of RO evaluations and a set $\Phi := \{\}$ of honest POPF indices. Sample $g \leftarrow [0, q'_h]$ and $\mathrm{A} \leftarrow (\{0,1\}^{3\kappa})^{\{0,1\}^{3\kappa} \times \mathbb{G}}$. Set $t_g := \bot$.

On $(\mathsf{Program}, \mathsf{sid})$ or $(\mathsf{Program}, \mathsf{sid}, \Sigma)$ from $\mathcal{F}$:

1. Choose $s \leftarrow \{0,1\}^{3\kappa}$ and $t \leftarrow \mathbb{G}$.
2. Add $(s,t)$ to $\Phi$ and send $(\mathsf{Program}, \mathsf{sid}, (s,t))$ to $\mathcal{F}$.

On $(\mathsf{Extract}, \mathsf{sid}, \phi^* = (s^*, t^*))$ from $\mathcal{F}$:

3. Abort if the guess $t_g$ is wrong. There are two cases:
   - A. If $H'(x, t^*)$ has been queried for some $x$, abort if $t_g \neq t^*$.
   - B. Otherwise, abort if $g \neq 0$.
4. Search for the anchor query "$h' = H'(x^*, t^*)$" $\in T_{\mathrm{RO}}$, which is the query satisfying one of these conditions:
   - A. There is an earlier query "$h^* = H(x^*, r^*)$" $\in T_{\mathrm{RO}}$ such that $r^* = s^* \oplus h'$.
   - B. There is an earlier query "$h'_\Phi = H'(x^*, t_\Phi)$" $\in T_{\mathrm{RO}}$, for some $(s_\Phi, t_\Phi) \in \Phi$, such that $s^* \oplus h' = s_\Phi \oplus h'_\Phi$.
5. If there is no anchor query, choose an arbitrary $x^* \in \mathcal{X}$.
6. Send $(\mathsf{Extract}, \mathsf{sid}, x^*, \mathrm{A}(s,t))$ to $\mathcal{F}$.

On $(\mathsf{Eval}, \mathsf{sid}, \phi = (s,t), x)$ from $\mathcal{F}$:

7. Evaluate $y := H(x, s \oplus H'(x,t)) \cdot t$, using the $\mathcal{F}_{\mathrm{RO}}$ interface defined below.
8. Send $(\mathsf{Eval}, \mathsf{sid}, y)$ to $\mathcal{F}$.

On $H(x,r)$ from $P$ aimed at $\mathcal{F}_{\mathrm{RO}}$:

9. Find a query "$h = H(x,r)$" $\in T_{\mathrm{RO}}$, or determine $h$ using to the following three cases if none exists:
   - A. If there exist $(s,t) \in \Phi$ and "$h' = H'(x,t)$" $\in T_{\mathrm{RO}}$ such that $r = s \oplus h'$, then send $(\mathsf{Eval}, \mathsf{sid}, (s,t), x)$ to $\mathcal{F}$. On response $(\mathsf{Eval}, \mathsf{sid}, y)$ from $\mathcal{F}$, compute $h := y/t$.
   - B. Otherwise if $t_g \neq \bot$ and there is a query "$h' = H'(x, t_g)$" $\in T_{\mathrm{RO}}$, then set $s_g := r \oplus h'$. Query $y := R(\mathrm{A}(s_g, t_g), x)$, and compute $h := y/t_g$.
   - C. Otherwise sample $h \in \mathbb{G}$.

   In all cases, append "$h = H(x,r)$" to $T_{\mathrm{RO}}$.
10. Return $h$ to $P$.

On $H'(x,t)$ from $P$ aimed at $\mathcal{F}_{\mathrm{RO}}$:

11. If $g \neq 0$ and $t$ is the $g$-th unique value that appears in such a query, set $t_g := t$.
12. Find a query "$h' = H'(x,t)$" $\in T_{\mathrm{RO}}$, or if none exists, sample $h' \leftarrow \{0,1\}^{3\kappa}$ and append "$h' = H'(x,t)$" to $T_{\mathrm{RO}}$.
13. Return $h'$ to $P$.

At the end of the experiment:

14. If $\mathsf{Extract}$ has not been called, abort unless $g = 0$.

Figure 13: POPF pseudo-simulator $\mathcal{S}_{\mathrm{pseudo}}$.

Parameters:

- Ideal functionalities $\mathcal{F}_0, \mathcal{F}_1$.
- Simulator $\mathcal{S}$.

Global variables:

- Lists $T_{\text{RO}}, T_R$ of RO evaluations.
- The extraction target $\phi^* = (s^*, t^*)$.
- A randomly sampled key $\alpha^* \leftarrow \{0,1\}^{3\kappa}$.

Run the simulator $\mathcal{S}$, delivering its messages with $\mathcal{F}_1$ as normal. Communication of $\mathcal{S}$ with $\mathcal{A}$ and corrupted parties depends on which ideal functionality $\mathcal{S}$ is simulating. Messages sent as $\mathcal{F}_0$ are delivered normally, while $\mathcal{F}_{\text{POPF}}$ messages and $R$ lookups are handled as follows.

On queries $(\mathsf{Program}, \mathsf{sid})$, $(\mathsf{Program}, \mathsf{sid}, A)$, $(\mathsf{Extract}, \mathsf{sid}, \phi)$, $(\mathsf{Eval}, \mathsf{sid}, \phi, x)$, $H(x, r)$, or $H'(x, t)$:

1. Handle the query as in $\mathcal{S}_{\text{pseudo}}$, except:
   - A. In $\mathsf{Extract}$, skip the abort (step 3), and return $\alpha^*$ instead of $\mathrm{A}(s^*, t^*)$.
   - B. In $H(x, r)$, remove case 9B to avoid querying $R$.
2. At the end of the experiment, skip the abort (step 14).

On oracle lookup $R(\alpha, x)$ from $\mathcal{S}$:

3. If there is a tuple $(\alpha, x, r) \in T_R$, return $r$.
4. Pick $r$ according to two cases:
   - A. If $\alpha = \alpha^*$ and $x \neq x^*$, evaluate $r := H(x, s^* \oplus H'(x, t^*)) \cdot t^*$.
   - B. Otherwise, sample $r \leftarrow \mathbb{G}$.
5. Add $(\alpha, x, r)$ to $T_R$.
6. Return $r$.

Figure 14: POPF simulator $\mathcal{S}_{\text{POPF}}$.

real world. The probability $\frac{1}{q'_h+1}$ was chosen to make the abort occur with equal probability between the two worlds, so that they are indistinguishable.

3. Pseudo-ideal world: $\mathcal{F}_1 \diamond \mathcal{S} \diamond \mathcal{S}_{\text{pseudo}}$. This is indistinguishable from the pseudo-intermediate world, since $\pi$ realizes $\mathcal{F}_1$.
4. Ideal world with abort: $\frac{1}{q'_h+1}(\mathcal{F}_1 \diamond \mathcal{S}_{\text{POPF}})$. This step has no analogy with the UC theorem. The idea is to make some tweaks to $\mathcal{S}_{\text{pseudo}}$ so that the abort will no longer be necessary, and then combine it with $\mathcal{S}$. While we cannot prove that $\pi_{\text{POPF}}$ realizes $\mathcal{F}_{\text{POPF}}$ due to the abort in $\mathcal{S}_{\text{pseudo}}$, we can remove the abort *in the context of combined protocols*, because $\mathcal{S}_{\text{POPF}}$ can now program $\mathcal{S}_{\text{pseudo}}$'s queries to the external RO $R$.

The proof concludes by noting that, if the environment has advantage $\mathsf{Adv}_{\text{POPF-abort}}$ of distinguishing the real world with abort from the ideal world with abort, then its actual advantage of distinguishing the real world from the ideal world (without abort) is upper-bounded as $\mathsf{Adv}_{\text{POPF}} \leq (q'_h + 1)\mathsf{Adv}_{\text{POPF-abort}}$.

**Bad events.** Even if $\mathcal{S}_{\text{pseudo}}$ does not abort, its simulation is still not perfect, due to the possibility of some bad events. It is more convenient to exclude the bad events from the beginning of the proof (i.e., in the real world). Formally, this can viewed as modifying the real world so that if any of these bad events occur then it will reset, and start running the ideal world from the start instead. Note, however, that these events will be defined using $\Phi$ and $\phi^* = (s^*, t^*)$, which are defined in the ideal functionality, not the real world. But they *can* all be observed by the honest parties (and so by the

environment) in the real world: $\Phi$ is the set of all outputs generated by the ideal functionality's Program interface, which is only run by honest parties as it is never called by the simulator, and $\phi^*$ is the first index not in $\Phi$ that is passed to Eval by an honest party. In the real world, we define them to match these observables.

Below we list these bad events:

- $\mathsf{Bad}_1$: When (Program, ...) queries $h = H(x^*, r)$ and $h' = H'(x^*, t)$, the inputs overlaps with previous $H$ and $H'$ queries. That is, $h$ and $h'$ are not freshly random.
- $\mathsf{Bad}_2$: When (Program, ...) computes $(s, t)$, there is some $(s', t) \in \Phi$; or if $(s^*, t^*)$ is defined, $t = t^*$.
- $\mathsf{Bad}_3$: When (Program, ...) is queried, returning $(s, t)$, look through the queries of the form "$H(x, r)$" in $\mathcal{F}_{\mathrm{RO}}$'s set $T_{\mathrm{RO}}$ from before this Program query. That is, exclude the $H$ query made during Program. The bad event occurs if there is a corresponding entry "$s \oplus r = H'(x, t)$" $\in T_{\mathrm{RO}}$, or if this entry is later added to $T_{\mathrm{RO}}$.
- $\mathsf{Bad}_4$: The honest POPF used by a given RO query is ambiguous. That is, for some "$H(x, r)$" $\in T_{\mathrm{RO}}$, there are distinct $(s, t), (s', t') \in \Phi$ such that both "$s \oplus r = H'(x, t)$" and "$s' \oplus r = H'(x, t')$" are in $T_{\mathrm{RO}}$.
- $\mathsf{Bad}_5$: The anchor query is not unique. That is, after all queries have been made, there are two distinct queries "$H'(x_0, t)$", "$H'(x_1, t)$" $\in T_{\mathrm{RO}}$ that both satisfy the conditions of being an anchor query.

Let Bad be the disjunction of these bad events. Assuming that Bad cannot occur can change the advantage by at most Pr[Bad]. Therefore, the environment's advantage will be bounded as $\mathsf{Adv}_{\mathrm{POPF}} \leq (q'_h + 1)\mathsf{Adv}_{\mathrm{POPF\text{-}abort}} + \Pr[\mathsf{Bad}]$.

**Detailed proof.** Let $\mathcal{Z}$ be the environment, and $\mathcal{A}$ be the adversary.

**Lemma 4.3.** *Assume $\mathcal{Z}$ issues $q_p$ Program commands, $q_h$ commands to $\mathcal{F}_{\mathrm{RO}}$ where the last bit of* sid *is* 0 *(i.e., $H$ queries), and $q'_h$ commands to $\mathcal{F}_{\mathrm{RO}}$ where the last bit of* sid *is* 1 *(i.e., $H'$ queries). Then in the real world,*

$$\Pr[\mathsf{Bad}] \leq \frac{(2q'_h + 3q_p + 1)q_p}{2|\mathbb{G}|} + \frac{q_p(q_p + 3)(q_h + q_p) + (q_h + q'_h + q_p)^2(q'_h + q_p)}{2^{3\kappa+1}}.$$

*Proof.* We bound the probability of each bad event, assuming that the previous bad events do not occur. All of our bad events are contained in unions of simpler bad events, such as sampling $t \leftarrow \mathbb{G}$ and finding that it equals a predetermined value $g \in \mathbb{G}$, which have obviously negligible probabilities. The sizes of these unions are polynomials in the number of executions of Program, $H$, and $H'$. Note, however, that the total number of executions of $H$ (resp. $H'$) is really $q_h + q_p$ (resp. $q'_h + q_p$), not $q_h$ (resp. $q'_h$), because every call to Program issues one query to each of $H$ and $H'$,

- $\mathsf{Bad}_1$: When the $h := H(x^*, r)$ query is executed by Program, the value $r$ is freshly sampled from $\{0,1\}^{3\kappa}$. There are $q_h + q_p$ previous executions of $H$, so $r$ overlaps a past query with probability at most $q_h/2^{3\kappa}$. Assuming that $r$ is distinct, the result $h$ is freshly random from $\mathbb{G}$. Then $H'(x^*, t)$ will also be distinct from all past queries, except with probability at most $q'_h/|\mathbb{G}|$, because $t = y^*/h$. That is, $t$ will be uniformly random in $\mathbb{G}$, independent from all past queries to $H'$, of which there are at most $q'_h + q_p$. Therefore, a union bound shows that

$$\Pr[\mathsf{Bad}_1] \leq \frac{(q_h + q_p)q_p}{2^{3\kappa}} + \frac{(q'_h + q_p)q_p}{|\mathbb{G}|}.$$

- $\mathsf{Bad}_2$: We first note that a pair $(s,t)$ is added to $\Phi$ only when a Program command is issued by $\mathcal{Z}$, so $|\Phi| \leq q_p$. The value $t$ is generated by $\pi_{\mathrm{POPF}}$ using an RO query as $t = y^*/H(x^*,r)$, and $H(x^*,r)$ is freshly random assuming that $\mathsf{Bad}_1$ does not occur. Therefore, $t \in \mathbb{G}$ is uniformly random and independent of $\Phi$, and the probability that $t$ is already in $\Phi$, i.e., there is a collision in the $t$ values in $\Phi$, is at most $\binom{q_p}{2}/|\mathbb{G}|$. Also, the probability that Program generates a pair $(s,t)$ such that $t = t^*$ is at most $q_p/|\mathbb{G}|$. Therefore,

$$\Pr[\mathsf{Bad}_2 \wedge \neg\mathsf{Bad}_1] \leq \frac{q_p^2 + q_p}{2|\mathbb{G}|}.$$

- $\mathsf{Bad}_3$: For $\mathsf{Bad}_3$ to occur, it must hold that $s \oplus r = H'(x,t)$ (or equivalently, $s = r \oplus H'(x,t)$) for some $s,t$ generated by Program and some $x,r$ with $H(x,r)$ *previously* queried by either $\mathcal{Z}$ or Program. There are at most $q_p$ $(s,t)$ pairs and at most $q_h + q_p$ $H$ queries; these two combined uniquely determine the values of $x$ and $t$, which in turn determine the $H'(x,t)$ query. For each pair $(s,t) \in \Phi$ and each $H(x,r)$ query, assuming that $\mathsf{Bad}_1$ does not occur, $s$ is a uniformly random string in $\{0,1\}^{3\kappa}$ independent of $r$ and $H'(x,t)$[21], so the probability that $s = r \oplus H'(x,t)$ is $1/2^{3\kappa}$. A union bound gives

$$\Pr[\mathsf{Bad}_3 \wedge \neg\mathsf{Bad}_1] \leq \frac{q_p(q_h + q_p)}{2^{3\kappa}}.$$

- $\mathsf{Bad}_4$: For $\mathsf{Bad}_4$ to occur, it must hold that $s \oplus r = H'(x,t)$ and $s' \oplus r = H'(x,t')$ for some distinct pairs $(s,t),(s',t')$ generated by Program and some $x,r$ with $H(x,r)$ queried by $\mathcal{Z}$ or Program. This implies that $s \oplus s' = H'(x,t) \oplus H'(x,t')$. Similarly to the analysis of $\mathsf{Bad}_3$, the query $H(x,r)$ and the POPF indexes $(s,t)$ and $(s',t')$ uniquely determine the values of $x,t,t'$, which in turn determine *both* the $H'(x,t)$ query and the $H'(x,t')$ query. Assuming that $\mathsf{Bad}_1$ and $\mathsf{Bad}_2$ do not occur, we have that $t \neq t'$, so $s, s', H'(x,t), H'(x,t')$ are mutually independent strings in $\{0,1\}^{3\kappa}$. Therefore,

$$\Pr[\mathsf{Bad}_4 \wedge \neg\mathsf{Bad}_1 \wedge \neg\mathsf{Bad}_2] \leq \frac{q_p(q_p-1)(q_h + q_p)}{2^{3\kappa+1}}.$$

  (The analysis of $\mathsf{Bad}_4$ is not covered by the analysis of $\mathsf{Bad}_3$, since here $r$ might depend on $s$.)

- $\mathsf{Bad}_5$: Recall that for an anchor query $(x,t)$, either of the followings must hold:
  - $r = s \oplus H'(x,t)$, where $H(x,r)$ was previously queried by $\mathcal{Z}$; or
  - $s \oplus H'(x,t) = s_\Phi \oplus H'(x,t_\Phi)$ for some $(s_\Phi, t_\Phi)$ sampled by an honest party,

  where $s,t$ are specified by the Extract command. Therefore, if there are two distinct anchor queries, then one of the followings must happen:
  - $r_0 \oplus H'(x_0,t) = s = r_1 \oplus H'(x_1,t)$ for some previous queries $H(x_0,r_0)$ and $H(x_1,r_1)$;
  - $r \oplus H'(x_0,t) = s = s_\Phi \oplus H'(x_1,t_\Phi) \oplus H'(x_1,t)$ for some previous $H(x_0,r)$ query and some $(s_\Phi,t_\Phi)$ sampled by an honest party; or
  - $s_{\Phi,0} \oplus H'(x_0,t_{\Phi,0}) \oplus H'(x_0,t) = s = s_{\Phi,1} \oplus H'(x_1,t_{\Phi,1}) \oplus H'(x_1,t)$ for some $(s_{\Phi,0},t_{\Phi,0})$ and $(s_{\Phi,1},t_{\Phi,1})$ sampled by an honest party.

[21] We can ignore the $H'(x^*,t)$ query made in Program, because the corresponding $H(x^*,r)$ query has been excluded.

Let us count the ways in which these sub-events can occur. All variables in the first sub-event are determined by the two $H$ queries and an $H'$ query; assuming $\mathsf{Bad}_2$ does not occur, all variables in the second sub-event are determined by the queries $H(x_0, r)$, $H'(x_0, t)$, and $H'(x_1, t_\Phi)$ (note that $s_\Phi$ is uniquely determined by $t_\Phi$); assuming $\mathsf{Bad}_2$ does not occur, all variables in the third sub-event are determined by the $H'(x_0, t)$, $H'(x_0, t_{\Phi,0})$, and $H'(x_1, t_{\Phi,1})$ queries. In all three sub-events, one of the $H'$ queries will come last, and so be uniformly random in $\{0,1\}^{3\kappa}$, independent of the $H$ queries and the other variables, and so will trigger the bad event with probability $2^{-3\kappa}$. Adding up, we get

$$\Pr[\mathsf{Bad}_5 \wedge \neg\mathsf{Bad}_2] \leq \frac{q_h^2(q'_h + q_p) + 2q_h(q'_h + q_p)^2 + (q'_h + q_p)^3}{2^{3\kappa+1}} = \frac{(q_h + q'_h + q_p)^2(q'_h + q_p)}{2^{3\kappa+1}}.$$

Summing up the five bad events above yields the lemma. □

**Lemma 4.4.** *Assume that $\mathcal{Z}$ makes $q_R$ queries to the global RO $R$, and* Bad *does not occur. Then $\mathcal{Z}$'s distinguishing advantage between the real world with abort and the pseudo-intermediate world is at most $(2q_R + q_h)q_h/2^{3\kappa+1}$.*

*Proof.* Consider the following hybrid argument:

*Hybrid 0:* This is the real world with abort. The environment $\mathcal{Z}$'s view is shown in Figure 15. Note that the experiment aborts with probability $q'_H/(q'_H + 1)$ (see steps 5 and 8), but otherwise behaves exactly as the same as the real world.

Sample $g \leftarrow [0, q'_H]$.

On $(\mathsf{Program}, \mathsf{sid}, x^*, y^*)$ (where $x^* \in \mathcal{X}$ and $y^* \in \mathbb{G}$):

1. Choose $r \leftarrow \{0,1\}^{3\kappa}$.
2. Compute $t := y^*/H(x^*, r)$.
3. Compute $s := r \oplus H'(x^*, t)$.
4. Output $(\mathsf{Program}, \mathsf{sid}, (s, t))$.

On $(\mathsf{Eval}, \mathsf{sid}, (s, t), x)$ (where $x \in \mathcal{X}$, $s \in \{0,1\}^{3\kappa}$, and $t \in \mathbb{G}$):

5. If this is the first Eval message for sid, abort if either $g \neq 0$ and $t$ is not the $g$-th unique value that appears in a query $H'(\cdot, t)$, or if $g = 0$ and $H'(\cdot, t)$ has been queried before.
6. Compute $y := H(x, s \oplus H'(x, t)) \cdot t$.
7. Output $(\mathsf{Eval}, \mathsf{sid}, y)$.

At the end of the experiment:

8. If Eval has not been called, abort unless $g = 0$.

Figure 15: $\mathcal{Z}$'s view in the real world with abort.

*Hybrid 1:* At the end of $(\mathsf{Program}, \mathsf{sid}, x^*, y^*)$, define $F_{(s,t)}(x^*) := y^*$; at the beginning of $(\mathsf{Eval}, \mathsf{sid}, (s, t), x)$, if $F_{(s,t)}(x)$ is already defined, then return $(\mathsf{Eval}, \mathsf{sid}, y)$ where $y := F_{(s,t)}(x)$; at the end of $(\mathsf{Eval}, \mathsf{sid}, (s, t), x)$, define $F_{(s,t)}(x) := y$.

Suppose that in hybrid 1, $F_{(s,t)}(x)$ is already defined when $\mathcal{Z}$ sends a command $(\mathsf{Eval}, \mathsf{sid}, (s, t), x)$. This means that there is a previous command $(\mathsf{Program}, \mathsf{sid}, x, y)$ with $t = y/H(x, r)$ and $s = r \oplus H'(x, t)$ for some $r \in \{0,1\}^{3\kappa}$. But then $y = F_{(s,t)}(x) = H(x, r) \cdot t = H(x, s \oplus H'(x, t)) \cdot t$, exactly as how $y$ is computed in hybrid 0. The only other difference between hybrids 0 and 1 is the bookkeeping of the $F$ function. Therefore, hybrids 0 and 1 are identical.

The following hybrids 2 and 3 consider the indices of honest POPFs $(s,t) \in \Phi$.

*Hybrid 2:* On $(\mathsf{Program}, \mathsf{sid}, x^*, y^*)$, choose $(s,t) \leftarrow \{0,1\}^{3\kappa} \times \mathbb{G}$ and output $(\mathsf{Program}, \mathsf{sid}, (s,t))$. Furthermore, compute $r := s \oplus H'(x^*, t)$; if $H(x^*, r)$ is queried, return $y^*/t$.

Assuming $\mathsf{Bad}_4$ does not occur, when $\mathcal{Z}$ queries $H(x^*, r)$, there do not exist two distinct pairs $(s,t), (s',t')$ such that $r = s \oplus H'(x^*,t) = s' \oplus H'(x^*,t')$, so the query can be answered unambiguously *if we only consider those queries defined by* Program *commands.* Furthermore, assuming $\mathsf{Bad}_3$ does not occur, it cannot happen that an $H$-query defined by a Program command has already been queried during a previous Eval command. It follows that $H$ queries can be answered unambiguously, and thus hybrid 2 is well-defined.

We can see that in both hybrids 1 and 2, the variables $s, t, r, h = H(x^*, r)$ satisfy the equations

$$s \oplus r = H'(x^*, t), \quad y^* = h \cdot t;$$

if we choose $(r, h) \leftarrow \{0,1\}^{3\kappa} \times \mathbb{G}$ then we get hybrid 1, whereas if we choose $(s,t) \leftarrow \{0,1\}^{3\kappa} \times \mathbb{G}$ then we get hybrid 2. So hybrids 1 and 2 are identical.

*Hybrid 3:* On $(\mathsf{Eval}, \mathsf{sid}, (s,t), x)$, if $F_{(s,t)}(x)$ is undefined and $(s,t) \in \Phi$, sample $y \leftarrow \mathbb{G}$, set $F_{(s,t)}(x) := y$, and return $(\mathsf{Eval}, \mathsf{sid}, y)$. Furthermore, compute $r := s \oplus H'(x,t)$; when $H(x,r)$ is queried, return $y/t$.

Similar to the analysis in hybrid 2, assuming neither $\mathsf{Bad}_3$ nor $\mathsf{Bad}_4$ occurs, hybrid 3 is well-defined. By an analysis similar to that in hybrid 2, hybrids 2 and 3 are identical.

The following hybrids 4–7 consider the "designated malicious index" $(s^*, t^*)$.

*Hybrid 4:* In step 5 of Figure 15 (note that the previous hybrids do not change this step), if the experiment does not abort, set $(s^*, t^*) := (s,t)$ and find the anchor query "$H'(x^*, t^*)$". If there is no anchor query, choose an arbitrary $x^* \in \mathcal{X}$.

Assuming $\mathsf{Bad}_5$ does not occur, the anchor query is uniquely defined, hence hybrid 4 is well-defined. The only difference between hybrids 3 and 4 is bookkeeping, so they are identical.

*Hybrid 5:* Sample $\mathrm{A} \leftarrow \left(\{0,1\}^{3\kappa}\right)^{\{0,1\}^{3\kappa} \times \mathbb{G}}$ at the beginning of the experiment. In the case that there is an "$h' = H'(x, t_g)$" query and then an "$H(x,r)$" query for some $x, r$ (where $x$ may or may not be equal to $x^*$), set $s_g := r \oplus h'$, $y := R(\mathrm{A}(s_g, t_g), x)$, and answer with $h := y/t_g$.

Let Query be the event that $\mathcal{Z}$ queries $R(\mathrm{A}(s_g, t_g), x))$, for some $s_g, t_g$ guessed as above. Each of $\mathcal{Z}$'s queries to $R$ specifies an $x$, so the $H'(x, t_g)$ query is fixed. There are at most $q_h$ queries to $H(x,r)$, so a union bound gives $\Pr[\mathsf{Query}] \leq q_R q_h / 2^{3\kappa}$. Let Collide be the event that two queries $\mathrm{A}(s_g, t_g)$, $\mathrm{A}(s'_g, t'_g)$ generate the same output; obviously, $\Pr[\mathsf{Collide}] \leq q_h^2 / 2^{3\kappa+1}$.

We now argue that in hybrid 5, if Query and Collide do not occur, then the $y$ values are independently random in $\mathcal{Z}$'s view for different $H(x,r)$ queries. Suppose $\mathcal{Z}$ makes two different queries $H(x_0, r_0), H(x_1, r_1)$. We have that (1) if $x_0 \neq x_1$, then the corresponding inputs to $R$ are different; (2) if $x_0 = x_1 = x$ but $r_0 \neq r_1$, then the corresponding $s$ values are $r_0 \oplus H'(x, t_g)$ and $r_1 \oplus H'(x, t_g)$ which are different, so the corresponding inputs to $R$ are different because Collide does not occur. Finally, since Query does not occur, $\mathcal{Z}$ does not make any of the corresponding $R$ queries. We conclude that all the outputs of $R$, namely the $y$ values, are independently random in $\mathcal{Z}$'s view.

We can see that in both hybrids 4 and 5, the variables $y, h$ satisfy the equation

$$y = h \cdot t_g;$$

if we choose $h \leftarrow \mathbb{G}$ then we get hybrid 4, whereas if we choose $y \leftarrow \mathbb{G}$ then we get hybrid 5 (except when Query or Collide occurs). Therefore, $\mathcal{Z}$'s distinguishing advantage between hybrids 4 and 5 is

at most $\Pr[\mathsf{Query}] + \Pr[\mathsf{Collide}] \leq (2q_R + q_h)q_h/2^{3\kappa+1}$.

*Hybrid 6:* In the condition in hybrid 4 — that is, when step 5 of Figure 15 runs, assuming it does not abort — if $F_{(s^*,t^*)}(x^*)$ is not already defined and $x \neq x^*$, then compute $y^* := H(x^*, s^* \oplus H'(x^*, t^*)) \cdot t^*$ and set $F_{(s^*,t^*)}(x^*) := y^*$ (in addition to what hybrid 4 already does). Note that the condition in hybrid 5 may be triggered while computing the $H$ output.

In hybrid 5, $F_{(s^*,t^*)}(x^*)$ is defined as $y^*$ when $\mathcal{Z}$ sends $(\mathsf{Eval}, \mathsf{sid}, (s^*, t^*), x^*)$, whereas in hybrid 6, $F_{(s^*,t^*)}(x^*)$ is defined as $y^*$ when $\mathcal{Z}$ sends $(\mathsf{Eval}, \mathsf{sid}, (s^*, t^*), x)$ for the first time for any $x$. This change does not affect $\mathcal{Z}$'s view.

*Hybrid 7:* On $(\mathsf{Eval}, \mathsf{sid}, (s^*, t^*), x)$, if $x \neq x^*$ and $F_{(s^*,t^*)}(x)$ is undefined, set $y := R(\alpha^*, (s^*, t^*), x)$ and $F_{(s^*,t^*)}(x) := y$, and return $(\mathsf{Eval}, \mathsf{sid}, y)$.

The difference between hybrids 6 and 7 is that an $(\mathsf{Eval}, \mathsf{sid}, (s^*, t^*), x)$ command is answered with $H(x, s^* \oplus H'(x, t^*)) \cdot t^*$ in hybrid 6 and $R(\alpha^*, (s^*, t^*), x)$ in hybrid 7. We consider two cases:

- If $\mathcal{Z}$ has queried $H(x, s^* \oplus H'(x, t^*))$:
  - If $\mathcal{Z}$ queried $H(x, s^* \oplus H'(x, t^*))$ and then $H'(x, t^*)$, this means that $H'(x, t^*)$ is an anchor query and thus $x = x^*$, so the change in hybrid 7 does not affect this case.
  - If $\mathcal{Z}$ queried $H'(x, t^*)$ and then $H(x, s^* \oplus H'(x, t^*))$, according to the description of hybrid 5, we have that $R(\alpha^*, (s^*, t^*), x) = H(x, s^* \oplus H'(x, t^*)) \cdot t^*$ since $t^* = t^g$ (because we haven't aborted), so $\mathcal{Z}$'s views in hybrids 6 and 7 are identical.
- If $\mathcal{Z}$ has not queried $H(x, s^* \oplus H'(x, t^*))$:
  - $H(x, s^* \oplus H'(x, t^*))$ appears in the experiment only when $\mathcal{Z}$ sends $(\mathsf{Eval}, \mathsf{sid}, (s_\Phi, t_\Phi), x)$ for some other $(s_\Phi, t_\Phi)$ such that $s^* \oplus H'(x, t^*) = s_\Phi \oplus H'(x, t_\Phi)$. But then $H'(x, t^*)$ is an anchor query and thus $x = x^*$, so the change in hybrid 7 does not affect this case.
  - If $H(x, s^* \oplus H'(x, t^*))$ does not appear in the experiment, then both $R(\alpha^*, (s^*, t^*), x)$ and $H(x, s^* \oplus H'(x, t^*))$ are random elements of $\mathbb{G}$, so $\mathcal{Z}$'s views in hybrids 6 and 7 are identical.

We conclude that $\mathcal{Z}$'s views in hybrids 6 and 7 are identical.

We now claim that hybrid 7 is identical to the pseudo-intermediate world. In both worlds, a $(\mathsf{Program}, \mathsf{sid}, x^*, y^*)$ command is answered with $(s, t) \leftarrow \{0,1\}^{3\kappa} \times \mathbb{G}$, and an $H'$ query is answered with a random string in $\{0,1\}^{3\kappa}$. For an $(\mathsf{Eval}, \mathsf{sid}, (s, t), x)$ command, in both worlds,

- If $F_{(s,t)}(x)$ is already defined, then the answer is $F_{(s,t)}(x)$. This can be seen from hybrid 1 and step 5 (without entering any of the sub-conditions) of Figure 10.
- If $F_{(s,t)}(x)$ is undefined and $(s, t) \in \Phi$, then the answer is a uniformly random element of $\mathbb{G}$. This can be seen from hybrid 3 and step 5A of Figure 10.
- If $F_{(s,t)}(x)$ is undefined and $(s, t) = (s^*, t^*)$, then if $x = x^*$, the answer is $H(x^*, s^* \oplus H'(x^*, t^*)) \cdot t^*$ as can be seen from hybrid 6 and step 4 of Figure 10; if $x \neq x^*$, the answer is $R(\alpha^*, (s^*, t^*), x)$ as can be seen from hybrid 7 and step 5B of Figure 10.
- If $F_{(s,t)}(x)$ is undefined, $(s, t) \in \Phi$ and $(s, t) \neq (s^*, t^*)$, then the answer is $H(x, s \oplus H'(x, t)) \cdot t$. This is unchanged throughout the hybrids and in the pseudo-intermediate world can be seen from step 5C of Figure 10.

Finally, for an $H(x,r)$ query, in both worlds,

- If $r = s \oplus H'(x,t)$ for some $(s,t) \in \Phi$, then the answer is $y/t = F_{(s,t)}(x)/t$. This can be seen from hybrid 3 and step 9A of Figure 13.
- If there is an "$h' = H'(x,t_g)$" query and then an "$H(x,r)$" query for some $x, r$, then the answer is $y/t_g = R(\alpha^*, x)/t_g$. This can be seen from hybrid 5 and step 9B of Figure 13.
- Otherwise the answer is a random element in $\mathbb{G}$. This is unchanged throughout the hybrids and in the pseudo-intermediate world can be seen from step 9C of Figure 13.

We conclude that hybrid 7 and the pseudo-intermediate world are identical. The only hybrid that generates a non-identical view is hybrid 5, so $\mathcal{Z}$'s distinguishing advantage between the real world with abort and the pseudo-intermediate world is at most $(2q_R + q_h)q_h/2^{3\kappa+1}$. This completes the proof. □

**Lemma 4.5.** *Assume that $\mathcal{Z}$ makes $q_R$ queries to the global RO $R$, and* Bad *does not occur. Then $\mathcal{Z}$'s distinguishing advantage between the pseudo-ideal world from the ideal world with abort is at most $3(q_h + q_R)^2/2^{3\kappa+1}$.*

*Proof.* Consider the following hybrid argument:

*Hybrid 0:* This is the pseudo-ideal world. Step 9B of the pseudo-simulator $\mathcal{S}_{\text{pseudo}}$ currently programs $H$ so that $H(x, H'(x,t_g) \oplus s_g) \cdot t_g = R(\mathrm{A}(s_g,t_g), x)$ holds for all $s_g$, whenever the $H'$ query is made before the $H$ query. In particular, this holds for $(s_g,t_g) = (s^*,t^*)$ and $x \neq x^*$, by the uniqueness of the anchor query (or else $\mathsf{Bad}_5$ would occur). When such an $H$ query is made for any $s_g$, $\mathcal{S}_{\text{pseudo}}$ computes $\alpha = \mathrm{A}(s_g,t_g)$ and $R(\alpha,x)$, and uses them to define the output of $H$. $R(\alpha,(s_g,t_g),x)$ is either a freshly random value here, or if it was previously queried (e.g., by $\mathcal{S}$) then it was sampled randomly when it was first queried.

*Hybrid 1:* Change to an equivalent distribution by swapping which of $H$ and $R$ is used to define the other. Instead of sampling $R$ and programming $H$ to match as with case 9B, let $H$ be sampled uniformly and program $R$ to match. However, there is no need to program $R(\alpha,x)$ other than when $\alpha = \alpha^*$ (i.e., $\alpha = \mathrm{A}(s^*,t^*)$), since no other evaluation of A is ever revealed by $\mathcal{S}_{\text{pseudo}}$. Therefore, we only program $R(\alpha^*,x)$ for $x \neq x^*$.

More precisely, let $H$ be defined as in the ideal world, by removing step 9B from $\mathcal{S}_{\text{pseudo}}$. Then whenever $R(\alpha^*,x)$ is evaluated (after $s^*$, $t^*$ and $\alpha^*$ have been defined), if $x \neq x^*$, compute $r = H(x, H'(x,t^*) \oplus s^*) \cdot t^*$ and program $R(\alpha^*,x) := r$. Steps 1B and 3–6 of Figure 14 are pseudocode for these two changes.

This change can only be noticed if either the $R$ evaluations are not all independently random, or if $\mathcal{Z}$ manages to make an $R(\alpha,x)$ query on some $\alpha = \mathrm{A}(s_g,t_g)$ not yet revealed by the simulator. That is, either there was a collision in A, or when $\mathcal{Z}$'s $R(\alpha,x)$ query was made either $\alpha \neq \alpha^*$ or $\alpha^*$ was not yet been returned by Extract. There are at most $q_h + q_R$ queries to A and to $R$, since A is only queried by $H$, and one of $R$ and $H$ queries the other.[22] Therefore, these events are upper bounded by $(q_h + q_R)^2/2^{3\kappa+1}$ and $(q_h + q_R)^2/2^{3\kappa}$, respectively, so $\mathcal{Z}$'s distinguishing advantage between hybrids 1 and 2 is at most $3(q_h + q_R)^2/2^{3\kappa+1}$.

*Hybrid 2:* Remove the RO A, and instead just sample $\alpha^* \leftarrow \{0,1\}^{3\kappa}$ and replace the query to $\mathrm{A}(s^*,t^*)$ with $\alpha^*$. The is equivalent because A is only called on input $(s^*,t^*)$.

[22] Before the hybrid, $H$ queries $R$, and after the hybrid $R$ queries $H$.

*Hybrid 3:* Notice that $g$ and $t_g$ are no longer used, except for the abort. This abort always has probability $q'_H/(q'_H+1)$, so we can remove $g$ and replace it with a simple abort with this probability.

*Hybrid 4:* Combine the higher-level protocol's simulator $\mathcal{S}$ with the modified pseudo-simulator $\mathcal{S}_{\text{pseudo}}$. Call their composition $\mathcal{S}_{\text{POPF}}$. We are now at the pseudo-ideal world.

The only hybrid that generates a non-identical view is hybrid 5, so $\mathcal{Z}$'s distinguishing advantage between the real world with abort and the pseudo-intermediate world is at most $3(q_h+q_R)^2/2^{3\kappa+1}$. This completes the proof. □

We can now put these lemmas together to get a proof of Thm. 4.2.

*Proof.* First, we bound $\mathsf{Adv}_{\text{POPF-abort}}$ using the hybrid argument outlined above. These hybrids are first those given in Lem. 4.4, then a single hybrid change for the security of $\pi$, and finally those in Lem. 4.5. Adding the advantages together, we get

$$\begin{aligned}\mathsf{Adv}_{\text{POPF-abort}} &\leq \frac{(2q_R+q_h)q_h}{2^{3\kappa+1}} + \mathsf{Adv}_\pi + \frac{3(q_h+q_R)^2}{2^{3\kappa+1}} \\ &= \mathsf{Adv}_\pi + \frac{3q_R^2+8q_hq_R+5q_h^2}{2^{3\kappa+1}}.\end{aligned}$$

Finally, put this together with Lem. 4.3 to get:

$$\begin{aligned}\mathsf{Adv}_{\text{POPF}} \leq& (q'_h+1)\mathsf{Adv}_{\text{POPF-abort}} + \Pr[\mathsf{Bad}] \\ \leq& (q'_h+1)\mathsf{Adv}_\pi + \frac{(q'_h+1)(3q_R^2+8q_hq_R+5q_h^2)}{2^{3\kappa+1}} + \frac{(2q'_h+3q_p+1)q_p}{2|\mathbb{G}|} \\ &+ \frac{q_p(q_p+3)(q_h+q_p)+(q_h+q'_h+q_p)^2(q'_h+q_p)}{2^{3\kappa+1}} \\ \leq& (q'_h+1)\mathsf{Adv}_\pi + \frac{(2q'_h+3q_p+1)q_p}{2|\mathbb{G}|} + \frac{(q_h+q'_h+q_R+q_p+1)^3}{2^{3\kappa}}.\end{aligned}$$

□

# 5 PAKE Protocol Based on POPF

In this section we present our main results, namely the UC-security analysis of EKE and OEKE using POPF. Concretely,

- In Sect. 5.2 we prove that EKE-PRF using POPF is UC-secure assuming the underlying KA protocol satisfies security, strong pseudorandomness, and pseudorandom non-malleability. Additional results about the "raw" version of EKE, including (1) EKE using IC is UC-secure, and (2) EKE using POPF realizes the weaker "PAKE with same password test" functionality $\mathcal{F}_{\text{PAKE-sp}}$ (both results are under the same assumptions on KA), are argued in Appx. D.
- In Sect. 5.3 we prove that OEKE using POPF is UC-secure assuming the underlying KA protocol satisfies security, pseudorandomness, pseudorandom non-malleability, and collision resistance. This covers the OEKE-PRF and OEKE-RO variants in existing works.

Our starting observation is that the first round of EKE and OEKE are exactly identical. This in particular means that several hybrids can be reused in both the EKE and OEKE proofs. To be more precise, there are several hybrids in the EKE and OEKE proofs where everything on the $P'$ side (recall $P'$ outputs first) is changed to random in certain cases, and the indistinguishability arguments are independent of the second round. To make our security proofs more concise and

modular, we abstract the first round into an ideal functionality $\mathcal{F}_{\text{PAKE-1r}}$, and prove that the first round of (O)EKE realizes $\mathcal{F}_{\text{PAKE-1r}}$; after that, we prove separately that the EKE and OEKE are secure in the $\mathcal{F}_{\text{PAKE-1r}}$-hybrid world. The main rationale behind this abstraction is that both the EKE and OEKE proofs contain a number of subtle hybrids and arguments, so any possibility of breaking them down into smaller pieces seems appealing (after all, this is the spirit of the UC framework itself).

### 5.1 The First-round Functionality and Protocol

**The functionality.** See Figure 16 for the UC functionality $\mathcal{F}_{\text{PAKE-1r}}$ representing the first-round of (O)EKE, parameterized by a specific underlying KA protocol $\mathsf{KA}$.

Parameters: KA protocol $\mathsf{KA} = (\mathsf{msg}_1, \mathsf{msg}_2, \mathsf{key}_1, \mathsf{key}_2)$.

- On input $(\mathsf{Program}, \mathsf{sid}, P, P', \mathsf{pw}, A)$ from $P$, send $(\mathsf{Program}, \mathsf{sid}, P, P')$ to $\mathcal{S}$. If this is the first $\mathsf{Program}$ message for $\mathsf{sid}$, then record $\langle \mathsf{Program}, P, P', \mathsf{pw}, A \rangle$.
- On input $(\mathsf{SampleResp}, \mathsf{sid}, P', P, \mathsf{pw}')$ from $P'$, send $(\mathsf{SampleResp}, \mathsf{sid}, P', P)$ to $\mathcal{S}$. If this is the first $\mathsf{SampleResp}$ message for $\mathsf{sid}$, then sample $b \leftarrow \mathcal{R}$ and record $\langle \mathsf{SampleResp}, P', P, \mathsf{pw}', b \rangle$.
- On $(\mathsf{Eval}, \mathsf{sid}, P, P', x)$ from $\mathcal{S}$, send $A$ to $\mathcal{S}$, where $A$ is defined as follows:
  - If there is a record $\langle \mathsf{Program}, P, P', \mathsf{pw}, A' \rangle$ with $\mathsf{pw} = x$, set $A := A'$.
  - Otherwise if there is a record $\langle \mathsf{Eval}, \mathsf{pw}, A'' \rangle$ with $\mathsf{pw} = x$, set $A := A''$.
  - Otherwise sample $a \leftarrow \mathcal{R}$ and set $A := \mathsf{msg}_1(a)$. Record $\langle \mathsf{Eval}, x, A \rangle$.
- On $(\mathsf{Deliver}, \mathsf{sid}, P, P', \mathsf{pw}^*, A^*)$ from $\mathcal{S}$, if there is a record $\langle \mathsf{SampleResp}, P', P, \mathsf{pw}', b \rangle$, and this is the first $\mathsf{Deliver}$ message for $\mathsf{sid}$, then output $(\mathsf{sid}, B, K')$ to $P'$, where $B$ and $K'$ are defined as follows:
  1. If $\mathsf{pw}^* = \perp$ and there is a record $\langle \mathsf{Program}, P, P', \mathsf{pw}, A \rangle$, overwrite $\mathsf{pw}^* := \mathsf{pw}$ and $A^* := A$.
  2. Next, if $\mathsf{pw}^* = \mathsf{pw}'$, then set $B := \mathsf{msg}_2(b, A^*)$ and $K' := \mathsf{key}_2(b, A^*)$. Else, sample $B \leftarrow \mathcal{M}_2$ and $K' \leftarrow \mathcal{K}$.

Figure 16: Ideal functionality $\mathcal{F}_{\text{PAKE-1r}}$ representing the first round of (O)EKE.

Recall that in the first round of both EKE and OEKE, party $P$ (with password $\mathsf{pw}$) computes its KA message $A$, programs $\phi$ such that $A = F_\phi(\mathsf{pw})$, and sends $\phi$ to $P'$. Then $P'$ (with password $\mathsf{pw}'$) evaluates $A' = F_\phi(\mathsf{pw}')$, samples randomness $b \leftarrow \mathcal{R}$, sends $B := \mathsf{msg}_2(b, A')$ to $P$, and outputs $K' := \mathsf{key}_2(b, A')$. (Note that the randomness $a$ that corresponds to $A$ is not used in the first round.) The man-in-the-middle adversary has the following capacity:

- The adversary can evaluate $F_\phi(x)$ for any (fresh) $x$ of its choice. If $x = \mathsf{pw}$ (which is the point at which $F_\phi$ is programmed) the result is $A$, otherwise the result is a random value.
- The adversary may or may not modify the $P$-to-$P'$ message $\phi$:
  - Suppose the adversary does not modify the message. Then if $\mathsf{pw} = \mathsf{pw}'$ (i.e., the passwords of $P$ and $P'$ match), $P'$ will send $B = \mathsf{msg}_2(b, A)$ to $P$ and output $K' = \mathsf{key}_2(b, A)$; if

$\mathsf{pw} \neq \mathsf{pw}'$, $P'$ will send $B = \mathsf{msg}_2(b, A')$ and output $K' = \mathsf{key}_2(b, A')$ where $A' = F_\phi(\mathsf{pw}')$ is a random value, so $B$ are $K'$ are pseudorandom.

– Suppose the adversary modifies the message $\phi$ to some other $\phi^*$, which incorporates a password guess $\mathsf{pw}^*$. Let $A^* = F_{\phi^*}(\mathsf{pw}^*)$; if $\mathsf{pw}^* = \mathsf{pw}'$ (i.e., the password guess is correct), $P'$ will send $B = \mathsf{msg}_2(b, A^*)$ to $P$ and output $K' = \mathsf{key}_2(b, A^*)$, and the adversary may compute $K'$ as $\mathsf{key}_1(a^*, B)$ (where $a^*$ is the randomness corresponding to $A^*$). If $\mathsf{pw}^* \neq \mathsf{pw}'$, $P'$ will send $B = \mathsf{msg}_2(b, A')$ and output $K' = \mathsf{key}_2(b, A')$ where $A' = F_{\phi^*}(\mathsf{pw}')$ is a random value, so $B$ are $K'$ are pseudorandom.

In the ideal world, the evaluation of $F_\phi(x)$ is modeled by the Eval command that defines the answer $A$ just as in the ideal world, except that if $\mathsf{pw} \neq x$ the answer is the "real" $\mathsf{msg}_1(a)$ instead of random. For the password test modeled by the Deliver command, the ideal adversary $\mathcal{S}$ may specify a $\mathsf{pw}^*$ and an $A^*$, and there are three cases:

- $\mathsf{pw}^* = \perp$ models the real-world scenario where the adversary does not modify the $P$-to-$P'$ message. Then if $\mathsf{pw} = \mathsf{pw}'$, the functionality sets $B = \mathsf{msg}_2(b, A)$ and $K' = \mathsf{key}_2(b, A)$. (Concretely, the functionality enters step 1 and redefines $\mathsf{pw}^* := \mathsf{pw}$ and $A^* := A$, and then enters step 2, checks that $\mathsf{pw}^* = \mathsf{pw}'$, and sets $B = \mathsf{msg}_2(b, A^*)$ and $K' = \mathsf{key}_2(b, A^*)$.)
- $\mathsf{pw}^* \neq \perp$ models the real-world scenario where the adversary modifies the $P$-to-$P'$ message and incorporates a password guess $\mathsf{pw}^*$ for $P'$. If $\mathsf{pw}^* = \mathsf{pw}'$, the functionality sets $B = \mathsf{msg}_2(b, A^*)$ and $K' = \mathsf{key}_2(b, A^*)$.
- Otherwise (i.e., $\mathsf{pw}^* = \perp$ and $\mathsf{pw} \neq \mathsf{pw}'$; or $\mathsf{pw}^* \neq \perp$ and $\mathsf{pw}^* \neq \mathsf{pw}'$) the functionality samples $B$ and $K'$ at random.

**The protocol.** See Figure 17 for the first-round protocol. Recall that the first round of all (O)EKE variants we analyze in this work is identical.

Parameters:
- POPF functionality $\mathcal{F}_{\text{POPF}}$ (see Figure 10) with domain $\{0,1\}^*$ and range $\mathcal{M}_1$.

On input $(\mathsf{Program}, \mathsf{sid}, P, P', \mathsf{pw}, A)$, party $P$ does the following:
1. Send $(\mathsf{Program}, \mathsf{sid}, \mathsf{pw}, A)$ to $\mathcal{F}_{\text{POPF}}$ and wait for response $(\mathsf{Program}, \mathsf{sid}, \phi)$.
2. Send $(\mathsf{sid}, \phi)$ to $P'$.

On input $(\mathsf{SampleResp}, \mathsf{sid}, P', P, \mathsf{pw}')$, party $P'$ samples $b \leftarrow \mathcal{R}$ and records $\langle \mathsf{pw}', b \rangle$.

On message $(\mathsf{sid}, \phi)$ from $P$, if there is a record $\langle \mathsf{pw}', b \rangle$ then party $P'$ does the following:
3. Send $(\mathsf{Eval}, \mathsf{sid}, \phi, \mathsf{pw}')$ to $\mathcal{F}_{\text{POPF}}$ and wait for response $(\mathsf{Eval}, \mathsf{sid}, A')$.
4. Compute $B := \mathsf{msg}_2(b, A')$ and $K' := \mathsf{key}_2(b, A')$.
5. Output $(\mathsf{sid}, B, K')$.

Figure 17: (O)EKE first-round protocol based on a POPF.

**Security analysis.**

**Lemma 5.1.** *Suppose that* KA *is a secure and pseudorandom KA protocol. Then the protocol in Figure 17 realizes* $\mathcal{F}_{\text{PAKE-1r}}$ *(Figure 16) in the* $\mathcal{F}_{\text{POPF}}$*-hybrid world.*

*Proof.* The following lemma will be useful while analyzing the security of the first-round protocol (a proof and further discussion is provided in Appx. A):

**Lemma 5.2.** *If a KA protocol is secure and pseudorandom then the following two distributions are indistinguishable:*

| | |
|---|---|
| $A \leftarrow \mathcal{M}_1$ | $A \leftarrow \mathcal{M}_1$ |
| $b \leftarrow \mathcal{R}$ | |
| $B := \mathsf{msg}_2(b, A)$ | $B \leftarrow \mathcal{M}_2$ |
| $K' := \mathsf{key}_2(b, A)$ | $K' \leftarrow \mathcal{K}$ |
| output $(A, B, K')$ | output $(A, B, K')$ |

We construct the simulator $\mathcal{S}$ in Figure 18. As standard in UC, we assume that the adversary $\mathcal{A}$ is "dummy" that merely passes all messages to and from the environment $\mathcal{Z}$. We use the following conventions: if $\mathcal{S}$ sends a message $m$ to $\mathcal{Z}$ that pretends to be from functionality $\mathcal{F}$ to $\mathcal{A}$, or from party $P$ to $P'$ and intercepted by $\mathcal{A}$, we abbreviate it as "send $m$ from $\mathcal{F}$ to $\mathcal{A}$ (or from $P$ to $P'$)" — although $\mathcal{A}$ does not exist in the ideal world. Similarly, if $\mathcal{S}$ receives a message $m$ from $\mathcal{Z}$ that instructs $\mathcal{A}$ to send $m$ to $\mathcal{F}$ or $P$, we abbreviate it as "on $m$ from $\mathcal{A}$ to $\mathcal{F}$ (or $P$)". These conventions are reused in later proofs.

On $(\mathsf{Program}, \mathsf{sid}, P, P')$ from $\mathcal{F}_{\text{PAKE-1r}}$:

1. Send $(\mathsf{Program}, \mathsf{sid})$ from $\mathcal{F}_{\text{POPF}}$ to $\mathcal{A}$.
2. On $(\mathsf{Program}, \mathsf{sid}, \phi)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{POPF}}$, send $(\mathsf{sid}, \phi)$ from $P$ to $P'$.

On $(\mathsf{SampleResp}, \mathsf{sid}, P', P)$ from $\mathcal{F}_{\text{PAKE-1r}}$ and $(\mathsf{sid}, \phi^*)$ from $\mathcal{A}$ to $P'$:

3. If $\phi^* = \phi$, send $(\mathsf{Deliver}, \mathsf{sid}, P, P', \perp, \perp)$ to $\mathcal{F}_{\text{PAKE-1r}}$.
4. Otherwise do the following steps:
   (1) Send $(\mathsf{Extract}, \mathsf{sid}, \phi^*)$ from $\mathcal{F}_{\text{POPF}}$ to $\mathcal{A}$.
   (2) On $(\mathsf{Extract}, \mathsf{sid}, \mathsf{pw}^*, \alpha^*)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{POPF}}$, send $(\mathsf{Eval}, \mathsf{sid}, \phi^*, \mathsf{pw}^*)$ from $\mathcal{F}_{\text{POPF}}$ to $\mathcal{A}$.
   (3) On $(\mathsf{Eval}, \mathsf{sid}, A^*)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{POPF}}$, send $(\mathsf{Deliver}, \mathsf{sid}, P, P', \mathsf{pw}^*, A^*)$ to $\mathcal{F}_{\text{PAKE-1r}}$.

Simulation of $\mathcal{F}_{\text{POPF}}$: run the code of $\mathcal{F}_{\text{POPF}}$ (Figure 10), except that when $\mathcal{A}$ queries $(\mathsf{Eval}, \mathsf{sid}, \phi, x)$ in step 5A, if $\phi$ has been generated (in step 2), send $(\mathsf{Eval}, \mathsf{sid}, P, P', x)$ to $\mathcal{F}_{\text{PAKE-1r}}$ and return $\mathcal{F}_{\text{PAKE-1r}}$'s response to $\mathcal{A}$.

Figure 18: Simulator $\mathcal{S}$ for the (O)EKE first-round protocol.

We now argue that $\mathcal{S}$ generates an ideal-world view that is indistinguishable from the real-world view. The proof goes by the following hybrid argument:

*Hybrid 0:* This is the real world. Recall that the passwords of $P$ and $P'$ are $\mathsf{pw}$ and $\mathsf{pw}'$, respectively; in the real world the flow of events is (for brevity, we omit $\mathsf{sid}$ in parties' messages and outputs below — same for later proofs):

- $P$ sends $\phi$ to $P'$ (intercepted by the man-in-the-middle adversary $\mathcal{A}$);
- $\mathcal{A}$ sends $\phi^*$ to $P'$;
- $P'$ outputs $(B, K')$.

Furthermore, $\mathcal{A}$ can evaluate $F_{\phi^*}(x)$ for $\phi^*, x$ of its choice by querying $(\mathsf{Eval}, \mathsf{sid}, \phi^*, x)$ to $\mathcal{F}_{\text{POPF}}$ and receiving answer $A$.

*Hybrid 1:* In the case that $\phi^* = \phi \wedge \mathsf{pw} \neq \mathsf{pw}'$, $P'$ outputs $B \leftarrow \mathcal{M}_2$ and $K' \leftarrow \{0,1\}^\kappa$.

In hybrid 0, $A' = F_{\phi^*}(\mathsf{pw}') = F_\phi(\mathsf{pw}')$ is uniformly random in $\mathcal{M}_1$, and $B = \mathsf{msg}_2(b, A')$ and $K' = \mathsf{key}_2(b, A')$. By Lem. 5.2, hybrids 0 and 1 are indistinguishable.

*Hybrid 2:* Consider the case that $\phi^* \neq \phi$ (i.e., $\mathcal{A}$ modifies the $P$-to-$P'$ message), and on $(\mathsf{Extract}, \mathsf{sid}, \phi^*)$ from $\mathcal{F}_{\text{POPF}}$, $\mathcal{A}$ replies with $(\mathsf{Extract}, \mathsf{sid}, \mathsf{pw}^*, \alpha^*)$ where $\mathsf{pw}^* \neq \mathsf{pw}'$. (Intuitively, $\phi^*$ contains a wrong password guess for $P'$.) Then sample $B \leftarrow \mathcal{M}_2$ and $K' \leftarrow \{0,1\}^\kappa$.

In hybrid 1 $K' = \mathsf{key}_2(b, A')$, where $A' = F_{\phi^*}(\mathsf{pw}')$ is determined via the following process: $\mathcal{F}_{\text{POPF}}$ sends $(\mathsf{Extract}, \mathsf{sid}, \phi^*)$ and receives $\mathcal{A}$'s response $(\mathsf{Extract}, \mathsf{sid}, \mathsf{pw}^*, \alpha^*)$, and later defines $A'$ as $R(\alpha^*, \mathsf{pw}')$ (i.e., $\mathcal{F}_{\text{POPF}}$ enters step 5B of Figure 10) which is uniformly random in $\mathcal{M}_1$. Thus, a reduction to Lem. 5.2, on input $(A', B, K')$, can randomly guess an $R$ query and program the answer as $A'$, and simulate other parts of hybrid 1 as usual and copy $\mathcal{Z}$'s output bit. The reduction loses a factor of $1/q$, where $q$ is the number of $R$ queries. By Lem. 5.2, hybrids 1 and 2 are indistinguishable.

*Hybrid 3:* When $\mathcal{A}$ queries $(\mathsf{Eval}, \mathsf{sid}, \phi, x)$ to $\mathcal{F}_{\text{POPF}}$, sample $a \leftarrow \mathcal{R}$ and answer with $\mathsf{msg}_1(a)$.

The only difference between hybrids 2 and 3 is that $F_\phi(x)$ (where $x \neq \mathsf{pw}$) is set to $A \leftarrow \mathcal{M}_1$ in hybrid 2 and $\mathsf{msg}_1(a)$ in hybrid 3. Note that $a$ is not used anywhere else in the entire experiment, so a straightforward reduction to the first pseudorandomness of $\mathsf{KA}$ shows that hybrids 2 and 3 are indistinguishable.

**Comparison between hybrid 3 and the ideal world.** We now argue that hybrid 3 is identical to the ideal world, where the simulator $\mathcal{S}$ as defined in Figure 18 interacts with $\mathcal{F}_{\text{PAKE-1r}}$. There are four cases:

- $\phi^* = \phi \wedge \mathsf{pw} = \mathsf{pw}'$: In hybrid 3 $A' = F_{\phi^*}(\mathsf{pw}') = F_\phi(\mathsf{pw}) = A$, so $B = \mathsf{msg}_2(b, A') = \mathsf{msg}_2(b, A)$ and $K' = \mathsf{key}_2(b, A') = \mathsf{key}_2(b, A)$. In the ideal world $\mathcal{S}$ sends $(\mathsf{Deliver}, \mathsf{sid}, P, P', \bot, \bot)$ to $\mathcal{F}_{\text{PAKE-1r}}$, which enters step 1 and overwrites $\mathsf{pw}^* := \mathsf{pw}$ and $A^* := A$; then $\mathcal{F}_{\text{PAKE-1r}}$ enters step 2 and $\mathsf{pw}^* = \mathsf{pw} = \mathsf{pw}'$, so $B = \mathsf{msg}_2(b, A^*) = \mathsf{msg}_2(b, A)$ and $K' = \mathsf{key}_2(b, A^*) = \mathsf{key}_2(b, A)$. So hybrid 3 and the ideal world are identical.

- $\phi^* = \phi \wedge \mathsf{pw} \neq \mathsf{pw}'$: In hybrid 3 $B \leftarrow \mathcal{M}_2$ and $K' \leftarrow \{0,1\}^\kappa$ (changed in hybrid 1). In the ideal world, $\mathcal{F}_{\text{PAKE-1r}}$ enters step 1 and overwrites $\mathsf{pw}^*$ and $A^*$ just as in the previous case, but then it enters step 2 and $\mathsf{pw}^* = \mathsf{pw} \neq \mathsf{pw}'$, so $B \leftarrow \mathcal{M}_2$ and $K' \leftarrow \{0,1\}^\kappa$. So hybrid 3 and the ideal world are identical.

- $\phi^* \neq \phi \wedge \mathsf{pw}^* = \mathsf{pw}'$: In hybrid 3 $A' = F_{\phi^*}(\mathsf{pw}')$, $B = \mathsf{msg}_2(b, A')$ and $K' = \mathsf{key}_2(b, A')$. In the ideal world $\mathcal{S}$ sends $(\mathsf{Deliver}, \mathsf{sid}, P, P', \mathsf{pw}^*, A^*)$ to $\mathcal{F}_{\text{PAKE-1r}}$, where $A^* = F_{\phi^*}(\mathsf{pw}^*) = F_{\phi^*}(\mathsf{pw}')$ is the value programmed by $\mathcal{A}$. Then $\mathcal{F}_{\text{PAKE-1r}}$ enters step 2 and $\mathsf{pw}^* = \mathsf{pw}'$, so $B = \mathsf{msg}_2(b, A^*) = \mathsf{msg}_2(b, A')$ and $K' = \mathsf{key}_2(b, A^*) = \mathsf{key}_2(b, A')$. So hybrid 3 and the ideal world are identical.

- $\phi^* \neq \phi \wedge \mathsf{pw}^* \neq \mathsf{pw}'$: In hybrid 3 $B \leftarrow \mathcal{M}_2$ and $K' \leftarrow \{0,1\}^\kappa$ (changed in hybrid 2). In the ideal world, $\mathcal{F}_{\text{PAKE-1r}}$ enters step 2 just as in the previous case, but $\mathsf{pw}^* \neq \mathsf{pw}'$, so $B \leftarrow \mathcal{M}_2$ and $K' \leftarrow \{0,1\}^\kappa$. So hybrid 3 and the ideal world are identical.

Finally, $\mathcal{A}$ can evaluate $F_{\phi^*}(x)$ by querying $\mathcal{F}_{\text{POPF}}$. Both hybrid 3 and the ideal world simply run the code $\mathcal{F}_{\text{POPF}}$, except that when $\phi^* = \phi \wedge x \neq \mathsf{pw}$, both hybrid 3 and the ideal world answers with $\mathsf{msg}_1(a)$ where $a \leftarrow \mathcal{R}$. So hybrid 3 and the ideal world are identical.

We conclude that hybrid 3 and the ideal world are identical in all cases. This completes the proof. □

**Remark 5.3.** *Lem. 5.1 essentially covers the hybrids that are reused in the proofs for EKE and OEKE, namely if (1) $\phi^* = \phi \wedge \mathsf{pw} \neq \mathsf{pw}'$ (i.e., the adversary passes the first message without modification, and the two parties' passwords do not match), or (2) $\phi^* \neq \phi \wedge \mathsf{pw}^* \neq \mathsf{pw}'$ (i.e., the*

*adversary modifies the first message, which contains a wrong password guess), it is safe to change everything on the $P'$ side to uniformly random (and independent of the $P$ side); furthermore, $F_\phi(x)$ (where $x \neq \mathsf{pw}$) can be set to $\mathsf{msg}_1(a)$ for $a \leftarrow R$, instead of a random $A \leftarrow \mathcal{M}_1$. Readers who would prefer to get rid of the $\mathcal{F}_{\text{PAKE}-1\text{r}}$ functionality and see a UC-security proof for (O)EKE as a monolith, can easily merge the hybrids above into the main proof, with the indistinguishability argument and the argument that the environment's view after the hybrids matches the ideal world essentially unchanged.*

## 5.2 The EKE Protocol

Parameters:

- EKE first round functionality $\mathcal{F}_{\text{PAKE-1r}}$ (Figure 16) for a KA protocol $\mathsf{KA}$.
- POPF functionality $\mathcal{F}_{\text{POPF}}$ (see Figure 10) with domain $\{0,1\}^*$ and range $\mathcal{M}_2$.
- A pseudorandom function $\mathsf{PRF}_K : \Phi \to \mathcal{K}$ with key space $\mathcal{K}$.

On input $(\mathsf{NewSession}, \mathsf{sid}, P, P', \mathsf{pw}, \text{“initiator”})$, party $P$ does the following:

1. Sample and record $a \leftarrow \mathcal{R}$, and compute $A := \mathsf{msg}_1(a)$.
2. Send $(\mathsf{Program}, sid, P, P', \mathsf{pw}, A)$ to $\mathcal{F}_{\text{PAKE-1r}}$.

On input $(\mathsf{NewSession}, \mathsf{sid}, P', P, \mathsf{pw}', \text{“respondent”})$, party $P'$ sends $(\mathsf{SampleResp}, sid, P, P', \mathsf{pw}')$ to $\mathcal{F}_{\text{PAKE-1r}}$.

On message $(\mathsf{sid}, B, K')$ from $\mathcal{F}_{\text{PAKE-1r}}$, party $P'$ does the following:

3. Send $(\mathsf{Program}, \mathsf{sid}, \mathsf{pw}', B)$ to $\mathcal{F}_{\text{POPF}}$ and wait for response $(\mathsf{Program}, \mathsf{sid}, \phi')$.
4. Send $(\mathsf{sid}, \phi')$ to $P$.
5. Output $(\mathsf{sid}, \mathsf{PRF}_{K'}(\phi'))$.

On message $(\mathsf{sid}, \phi')$ from $P'$, party $P$ does the following:

6. Send $(\mathsf{Eval}, \mathsf{sid}, \phi', \mathsf{pw})$ to $\mathcal{F}_{\text{POPF}}$ and wait for response $(\mathsf{Eval}, \mathsf{sid}, B)$.
7. Retrieve $a$ and compute $K := \mathsf{key}_1(a, B)$.
8. Output $(\mathsf{sid}, \mathsf{PRF}_K(\phi'))$.

Figure 19: EKE-PRF protocol, in the $(\mathcal{F}_{\text{POPF}}, \mathcal{F}_{\text{PAKE-1r}})$-hybrid model.

**Theorem 5.4.** *Suppose that $\mathsf{KA}$ is a correct, secure, strongly pseudorandom, and pseudorandom non-malleable KA protocol. Then the EKE-PRF protocol (Figure 19) realizes $\mathcal{F}_{\text{PAKE}}$ in the $(\mathcal{F}_{\text{POPF}}, \mathcal{F}_{\text{PAKE-1r}})$-hybrid world.*

*Proof.* The following lemma will be useful while analyzing the security of EKE-PRF (a proof and further discussion is provided in Appx. A):

**Lemma 5.5.** *If a KA protocol is secure and strongly pseudorandom then the following two distributions are indistinguishable:*

| | |
|---|---|
| $a \leftarrow \mathcal{R}$ | $a \leftarrow \mathcal{R}$ |
| $A := \mathsf{msg}_1(a)$ | $A := \mathsf{msg}_1(a)$ |
| $B \leftarrow \mathcal{M}_2$ | $B \leftarrow \mathcal{M}_2$ |
| $K := \mathsf{key}_1(a, B)$ | $K \leftarrow \mathcal{K}$ |
| output $(A, B, K)$ | output $(A, B, K)$ |

Initialize $U := \{\}$ as the set of $H$-evaluations.
Let $T = \{\}$ be the record of honest POPF evaluations as in $\mathcal{F}_{\text{POPF}}$.

To compute $H(x)$:

1. If there is an entry $(x, y) \in U$ return $y$.
2. Otherwise sample $y \leftarrow \mathcal{R}$, add $(x, y)$ to $U$ and return $y$.

On $(\mathsf{NewSession}, \mathsf{sid}, P, P', \text{“initiator”})$ from $\mathcal{F}_{\text{PAKE}}$:

3. Send $(\mathsf{Program}, \mathsf{sid}, P, P')$ from $\mathcal{F}_{\text{PAKE-1r}}$ to $\mathcal{A}$.

On $(\mathsf{NewSession}, \mathsf{sid}, P', P, \text{“respondent”})$ from $\mathcal{F}_{\text{PAKE}}$:

4. Send $(\mathsf{SampleResp}, \mathsf{sid}, P', P)$ from $\mathcal{F}_{\text{PAKE-1r}}$ to $\mathcal{A}$.

On $(\mathsf{Eval}, \mathsf{sid}, P, P', x)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{PAKE-1r}}$:

5. Compute $A = \mathsf{msg}_1(H(x))$ and return $A$ to $\mathcal{A}$.

On $(\mathsf{Deliver}, \mathsf{sid}, P, P', \mathsf{pw}^*, A^*)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{PAKE-1r}}$:

6. Send $(\mathsf{Program}, \mathsf{sid})$ from $\mathcal{F}_{\text{POPF}}$ to $\mathcal{A}$ and wait until $\mathcal{A}$ responds with $(\mathsf{Program}, \mathsf{sid}, \phi')$ to $\mathcal{F}_{\text{POPF}}$ such that there is no entry $(\phi', \cdot, \cdot) \in T$.
7. Send $(\mathsf{sid}, \phi')$ from $P'$ to $P$.
8. If $\mathsf{pw}^* = \bot$, send $(\mathsf{NewKey}, \mathsf{sid}, P', 0^\kappa)$ to $\mathcal{F}_{\text{PAKE}}$.
9. If $\mathsf{pw}^* \neq \bot$ do:
   (1) Send $(\mathsf{TestPwd}, \mathsf{sid}, P', \mathsf{pw}^*)$ to $\mathcal{F}_{\text{PAKE}}$.
   (2) Sample $b \leftarrow \mathcal{R}$ and compute $(K')^* := \mathsf{key}_2(b, A^*)$.
   (3) Send $(\mathsf{NewKey}, \mathsf{sid}, P', \mathsf{PRF}_{(K')^*}(\phi'))$ to $\mathcal{F}_{\text{PAKE}}$.

On $(\mathsf{sid}, (\phi')^*)$ from $\mathcal{A}$ to $P$:

10. If either (1) $\mathsf{pw}^* = \bot \wedge (\phi')^* = \phi'$ or (2) $\mathsf{pw}^* \neq \bot \wedge (\phi')^* = \phi'$ and the password guess on $\mathsf{Deliver}$ was incorrect, send $(\mathsf{NewKey}, \mathsf{sid}, P, 0^\kappa)$ to $\mathcal{F}_{\text{PAKE}}$.
11. If $\mathsf{pw}^* \neq \bot \wedge (\phi')^* = \phi'$ and the password guess on $\mathsf{Deliver}$ was correct:
    (1) Send $(\mathsf{TestPwd}, \mathsf{sid}, P, \mathsf{pw}^*)$ to $\mathcal{F}_{\text{PAKE}}$.
    (2) Compute $K^* = \mathsf{key}_1(H(\mathsf{pw}^*), \mathsf{msg}_2(b, A^*))$ and send $(\mathsf{NewKey}, \mathsf{sid}, P, \mathsf{PRF}_{K^*}(\phi'))$ to $\mathcal{F}_{\text{PAKE}}$.
12. Otherwise (i.e., if $(\phi')^* \neq \phi'$ or $\phi'$ is undefined because no $\mathsf{Deliver}$ message has been sent):
    (1) Send $(\mathsf{Extract}, \mathsf{sid}, (\phi')^*)$ from $\mathcal{F}_{\text{POPF}}$ to $\mathcal{A}$.
    (2) On $(\mathsf{Extract}, \mathsf{sid}, (\mathsf{pw}')^*, \alpha^*)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{POPF}}$, send $(\mathsf{Eval}, \mathsf{sid}, (\phi')^*, (\mathsf{pw}')^*)$ from $\mathcal{F}_{\text{POPF}}$ to $\mathcal{A}$.
    (3) On $(\mathsf{Eval}, \mathsf{sid}, (B')^*)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{POPF}}$, send $(\mathsf{TestPwd}, \mathsf{sid}, P, (\mathsf{pw}')^*)$, compute $K^* := \mathsf{key}_1(H((\mathsf{pw}')^*), (B')^*)$ and send $(\mathsf{NewKey}, \mathsf{sid}, P, \mathsf{PRF}_{K^*}((\phi')^*))$ to $\mathcal{F}_{\text{PAKE}}$.

Simulation of $\mathcal{F}_{\text{POPF}}$: run the code of $\mathcal{F}_{\text{POPF}}$ as in Figure 10, except that on $(\mathsf{Eval}, \mathsf{sid}, \phi', \mathsf{pw}^*)$, if the password guess on $\mathsf{Deliver}$ was correct return $B = \mathsf{msg}_2(b, A^*)$.

Figure 20: Simulator $\mathcal{S}$ for the EKE-PRF protocol.

| # | case addressed | change | properties used |
|---|---|---|---|
| 1 | $\mathsf{pw} = \mathsf{pw}'$ and $\mathcal{A}$ eavesdrops or re-encrypts | $K := K'$ | KA correctness |
| 2 | $\mathsf{pw} = \mathsf{pw}' \wedge \mathsf{pw}^* = \perp$ | $B, K'$ \$ | KA pseudorandom non-malleability |
| 3 | $(\phi')^*$ contains a wrong password guess or $\mathsf{pw} \neq \mathsf{pw}' \wedge (\phi')^* = \phi'$ | $K$ \$ | Lem. 5.5 (KA security / strong pseudorandomness) |
| 4 | $\mathsf{pw}^* \neq \perp \wedge \mathsf{pw} = \mathsf{pw}' \neq \mathsf{pw}^* \wedge (\phi')^* = \phi'$ | $K$ \$ | Lem. 5.5 (KA security / strong pseudorandomness) |
| 5 | $\mathcal{A}$ eavesdrops | $SK := SK'$ | none |
| 6 | all cases except $\mathsf{pw}^* = \mathsf{pw}'$ | $\mathsf{PRF}_{K'}$ \$ | PRF property |
| 7 | cases in hybrids 3 and 4 | $\mathsf{PRF}_K$ \$ | PRF property |
| 8 | $\mathcal{A}$ re-encrypts | $SK := \mathsf{PRF}_K((\phi')^*)$ | PRF property |
| 9 | | $K := \mathsf{key}_1(a, B)$ | Lem. 5.5 (KA security / strong pseudorandomness) |

Table 6: Summary of hybrids for EKE-PRF security. "\$" denotes "chosen at random from respective range". $SK, SK'$ denote the outputs of $P, P'$ respectively.

The simulator $\mathcal{S}$ is shown in Figure 20.

The proof goes by the following hybrid argument (see Table 6 for a summary):

*Hybrid 0:* This is the real world. Recall that the passwords of $P$ and $P'$ are $\mathsf{pw}$ and $\mathsf{pw}'$, respectively; the flow of events is:

- $P$ tells $\mathcal{F}_{\text{PAKE-1r}}$ to $\mathsf{Program}$ a function mapping $\mathsf{pw} \mapsto A$ and send it to $P'$;
- It is intercepted by the man-in-the-middle adversary $\mathcal{A}$, who possibly modifies it to program $\mathsf{pw}^* \mapsto A^*$ instead (intuitively $\mathsf{pw}^*$ is $\mathcal{A}$'s guess for the password of $P'$, $\mathsf{pw}'$);
- $\mathcal{A}$ tells $\mathcal{F}_{\text{PAKE-1r}}$ to $\mathsf{Deliver}$ the function to $P'$;
- $P'$ tells $\mathcal{F}_{\text{PAKE-1r}}$ to $\mathsf{SampleResp}$, which generates $B, K'$;
- $P'$ tells $\mathcal{F}_{\text{POPF}}$ to program a function $F_{\phi'}$ mapping $\mathsf{pw}' \mapsto B$;
- $P'$ outputs $\mathsf{PRF}_{K'}(\phi')$ and sends $\phi'$ to $P$;
- It is again intercepted by $\mathcal{A}$, who possibly modifies it to $(\phi')^*$ representing a function $F_{(\phi')^*}$ programmed on $(\mathsf{pw}')^* \mapsto (B')^*$ instead (intuitively $(\mathsf{pw}')^*$ is $\mathcal{A}$'s guess for the password of $P$, $\mathsf{pw}$);
- $\mathcal{A}$ sends $(\phi')^*$ to $P$;
- $P$ computes the key exchange message $B^* = F_{(\phi')^*}(\mathsf{pw})$ and uses it to output $\mathsf{PRF}_K((\phi')^*)$.

Note that there are three $B$ values here: $B = F_\phi(\mathsf{pw}')$ is computed by $P'$, $(B')^* = F_{(\phi')^*}((\mathsf{pw}')^*)$ is computed by $\mathcal{A}$, and $B^* = F_{(\phi')^*}(\mathsf{pw})$ is computed by $P$.

Throughout the proof, when $(\phi')^* \neq \phi'$ (i.e., $\mathcal{A}$ modifies the $P'$-to-$P$ message), and on

$(\mathsf{Extract}, \mathsf{sid}, (\phi')^*)$ from $\mathcal{F}_{\text{POPF}}$, $\mathcal{A}$ replies with $(\mathsf{Extract}, \mathsf{sid}, (\mathsf{pw}')^*, \alpha^*)$, we call this case "$(\phi')^*$ contains the correct password guess" if $(\mathsf{pw}')^* = \mathsf{pw}$, and "$(\phi')^*$ contains a wrong password guess" if $(\mathsf{pw}')^* \neq \mathsf{pw}$. (Note that we view $(\phi')^* = \phi'$ as $(\phi')^*$ contains no password guess, neither correct nor wrong.)

*Hybrid 1:* In the case that $\mathsf{pw} = \mathsf{pw}' \wedge \mathsf{pw}^* = \perp \wedge B^* = B$ and $(\phi')^*$ does not contain a wrong password guess, $P$ sets $K := K'$ instead of $K := \mathsf{key}_1(a, B^*)$. We have two sub-cases here:

- $\mathsf{pw} = \mathsf{pw}' \wedge \mathsf{pw}^* = \perp \wedge (\phi')^* = \phi'$ (which implies $B^* = B$ as $B^* = F_{(\phi')^*}(\mathsf{pw}) = F_{\phi'}(\mathsf{pw}') = B$). Intuitively this means that $\mathcal{A}$ merely eavesdrops.
- $\mathsf{pw} = \mathsf{pw}' \wedge \mathsf{pw}^* = \perp \wedge (\phi')^* \neq \phi' \wedge B^* = B$. Intuitively this means that $\mathcal{A}$ re-encrypts.

By the syntax of $\mathcal{F}_{\text{PAKE-1r}}$, $B = \mathsf{msg}_2(b, A)$ and $K' = \mathsf{key}_2(b, A)$, where $A = \mathsf{msg}_1(a)$ according to protocol description (step 2). By the correctness of $\mathsf{KA}$,

$$\mathsf{key}_1(a, B^*) = \mathsf{key}_1(a, B) = \mathsf{key}_1(a, \mathsf{msg}_2(b, \mathsf{msg}_1(a))) = \mathsf{key}_2(b, \mathsf{msg}_1(a)) = K'$$

with overwhelming probability, so hybrid 1 is (statistically) indistinguishable from hybrid 0.

*Hybrid 2:* In the case that $\mathsf{pw} = \mathsf{pw}' \wedge \mathsf{pw}^* = \perp$, $P'$ samples $B \leftarrow \mathcal{M}_2$ and $K' \leftarrow \mathcal{K}$.

The difference between hybrids 1 and 2 is that in hybrid 1, $B = \mathsf{msg}_2(b, A)$ and $K' = \mathsf{key}_2(b, A)$, whereas in hybrid 2, $B \leftarrow \mathcal{M}_2$ and $K' \leftarrow \mathcal{K}$. We construct a reduction $\mathcal{R}$ to the pseudorandom non-malleability of $\mathsf{KA}$: $\mathcal{R}$, given $(A, B, K')$, can simulate the experiment up to when $P'$ outputs — which includes $\mathcal{A}$'s access to $\mathcal{F}_{\text{PAKE-1r}}$, as well as $B$ and $K'$ — without knowing $a$ or $b$. (Note that in $\mathcal{F}_{\text{PAKE-1r}}$, $b$ is used only when computing $B$ and $K'$.) $\mathcal{R}$ simulates the second message $\phi'$ as expected, by programming $F_{\phi'}(\mathsf{pw}') = B$ into the POPF. When $\mathcal{A}$ sends $(\phi')^*$, $\mathcal{R}$ checks whether $B^* = F_{(\phi')^*}(\mathsf{pw})$ equals $B = F_{\phi'}(\mathsf{pw}') = F_{\phi'}(\mathsf{pw})$, and proceeds as follows:

- If $B^* \neq B$, $\mathcal{R}$ outputs $B^*$ to the pseudorandom non-malleability challenger and receives $K$, outputting $\mathsf{PRF}_K((\phi')^*)$ to $\mathcal{Z}$.
- If $B^* = B$, the pseudorandom non-malleability experiment prohibits $\mathcal{R}$ from outputting $B^*$. Instead, $\mathcal{R}$ checks whether $(\phi')^*$ contains a wrong password guess by checking that $(\phi')^* \neq \phi'$ and sending the appropriate $\mathsf{Extract}$ query from $\mathcal{F}_{\text{POPF}}$ to $\mathcal{A}$. If $(\phi')^*$ does contain a wrong password guess, $\mathcal{R}$ aborts. If not, $\mathcal{B}$ picks some other element of $\mathcal{M}_2$ to output to the pseudorandom non-malleability challenger (just to finish the experiment) and outputs $\mathsf{PRF}_{K'}((\phi')^*)$ to $\mathcal{Z}$ (note that in hybrid 1 $P$ was changed to use $K = K'$ in this case, so the correct value for $P$ is still output to $\mathcal{Z}$).

Finally, $\mathcal{R}$ copies $\mathcal{Z}$'s output bit.

We can see that if $B = \mathsf{msg}_2(b, A)$ and $K' = \mathsf{key}_2(b, A)$ then $\mathcal{R}$ simulates hybrid 1, and if $B \leftarrow \mathcal{M}_2$ and $K' \leftarrow \mathcal{K}$ then $\mathcal{R}$ simulates hybrid 2, as long as we don't have that $B^* = B$ and $(\phi')^*$ contains a wrong password guess. If that latter case occurs, $F_{(\phi')^*}$ is programmed on some $(\mathsf{pw}')^* \neq \mathsf{pw}$, therefore $B^* = F_{(\phi')^*}(\mathsf{pw}) = R(\alpha^*, \mathsf{pw})$ for some $\alpha^*$ (i.e., $\mathcal{F}_{\text{POPF}}$ enters step 5B of Figure 10), i.e., $B^*$ is a uniformly random element of $\mathcal{M}_2$ that is independent of $B$. So $B = B^*$ happens with probability $1/|\mathcal{M}_2|$, which is negligible when $\mathsf{KA}$ is a correct and secure KA protocol.[23]

[23] An adversary $\mathcal{A}$ may break the security of a correct $\mathsf{KA}$ with probability close to $1/|\mathcal{M}_2|$ as follows. Given $A = \mathsf{msg}_1(a), B = \mathsf{msg}_2(b, A)$ with $a, b \leftarrow \mathcal{R}$, $\mathcal{A}$ samples $b' \leftarrow \mathcal{R}$ and computes $B' = \mathsf{msg}_2(b', A)$. If $B = B'$ then with overwhelming probability $\mathsf{key}_1(a, B) = \mathsf{key}_1(a, B') = \mathsf{key}_2(b', A)$ by correctness, therefore $\mathcal{A}$ can distinguish $\mathsf{key}_1(a, B)$ from random. It is well-known that if $\mathbb{H}_\alpha(X)$ is the Rényi entropy of a random variable $X$ one has $\mathbb{H}_\alpha(X)$ non-increasing in $\alpha$. In particular, $\mathbb{H}_2(X) \leq \mathbb{H}_0(X)$: setting $X = [\mathsf{msg}_2(b, A) : b \leftarrow \mathcal{R}]$ gives the inequality $\Pr[B = B'] \geq 1/|\mathcal{M}_2|$.

We conclude that $\mathcal{R}$'s distinguishing advantage (in the pseudorandom non-malleability experiment for $\mathsf{KA}$) is negligibly smaller than $\mathcal{Z}$'s distinguishing advantage between hybrids 1 and 2, so hybrids 1 and 2 are indistinguishable.

*Hybrid 3:* In the case that $(\phi')^*$ contains a wrong password guess or $\mathsf{pw} \neq \mathsf{pw}' \wedge (\phi')^* = \phi'$, $P$ samples $K \leftarrow \mathcal{K}$.

In hybrid 2,

- If $(\phi')^*$ contains a wrong password guess, then as we have just argued under hybrid 2, $B^* = R(\alpha^*, \mathsf{pw})$.
- If $\mathsf{pw} \neq \mathsf{pw}' \wedge (\phi')^* = \phi'$ we have $B^* = F_{(\phi')^*}(\mathsf{pw}) = F_{\phi'}(\mathsf{pw})$. Since $\phi'$ was programmed on $\mathsf{pw}' \neq \mathsf{pw}$, $B^*$ is again defined as $R(\alpha^*, \mathsf{pw})$.

Either way, $B^* = R(\alpha^*, \mathsf{pw})$ is a uniformly random element of $\mathcal{M}_2$. This means that in hybrid 2 $K = \mathsf{key}_1(a, B^*)$ for $A = \mathsf{msg}_1(a)$ and $B^* \leftarrow \mathcal{M}_2$, whereas in hybrid 3 $K \leftarrow \mathcal{K}$. Thus, a reduction to Lem. 5.5, on input $(A, B^*, K)$, can randomly guess an $R$ query and program the answer as $B^*$, and simulate the other parts of hybrid 2 as usual (using $A$ and $K$ for $P$), copying $\mathcal{Z}$'s output bit. The reduction loses a factor of $1/q$, where $q$ is the number of $R$ queries. Since $\mathsf{KA}$ is secure and strongly pseudorandom, by Lem. 5.5, hybrids 2 and 3 are indistinguishable.

*Hybrid 4:* In the case that $\mathsf{pw}^* \neq \perp \wedge \mathsf{pw} = \mathsf{pw}' \neq \mathsf{pw}^* \wedge (\phi')^* = \phi'$, $P$ samples $K \leftarrow \mathcal{K}$.

The change in this hybrid is the same as in hybrid 3 ($P$ samples $K \leftarrow \mathcal{K}$ instead of $K := \mathsf{key}_1(a, B^*)$), but the argument for indistinguishability is different. In this case $B^* = F_{(\phi')^*}(\mathsf{pw}) = F_{\phi'}(\mathsf{pw}') = B$, so $\mathcal{F}_{\text{POPF}}$ does not enter step 5B and use $R$ to compute $B^*$. However, by the syntax of $\mathcal{F}_{\text{PAKE-1r}}$, when $\mathsf{pw}^* \neq \mathsf{pw}'$ we still have $B \leftarrow \mathcal{M}_2$. Therefore the reduction to Lem. 5.5 under hybrid 3 still works, except that here it does not need to guess an $R$ query and lose a factor of $q$.

**Summary of hybrid 4.** Let us pause a bit and summarize the changes we have made so far. On the $P'$ side, we have changed $B$ and $K'$ in some cases; on the $P$ side, we have changed $K$ in some cases. See Tables 7 and 8 for a summary.

| # | case | $B, K'$ definitions | according to... |
|---|---|---|---|
| 1 | $\mathsf{pw} = \mathsf{pw}' \wedge \mathsf{pw}^* = \perp$ | $B, K'$ \$ | hybrid 2 |
| 2 | $\mathsf{pw} \neq \mathsf{pw}' \wedge \mathsf{pw}^* = \perp$ | $B, K'$ \$ | $\mathcal{F}_{\text{PAKE-1r}}$ syntax |
| 3 | $\mathsf{pw}^* = \mathsf{pw}$ | $B = \mathsf{msg}_2(b, A^*)$<br>$K' = \mathsf{key}_2(b, A^*)$ | $\mathcal{F}_{\text{PAKE-1r}}$ syntax |
| 4 | $\mathsf{pw}^* \neq \perp \wedge \mathsf{pw}' \neq \mathsf{pw}$ | $B, K'$ \$ | $\mathcal{F}_{\text{PAKE-1r}}$ syntax |

Table 7: Definitions of $B$ and $K'$ in hybrid 4

| # | case | $K$ definition | according to... |
|---|---|---|---|
| 1 | $\mathsf{pw} = \mathsf{pw}' \wedge \mathsf{pw}^* = \perp \wedge (\phi')^* = \phi'$ (eavesdropping case) | $K = K'$ | hybrid 1 |
| 2 | $\mathsf{pw}^* = \mathsf{pw} = \mathsf{pw}' \wedge (\phi')^* = \phi'$ | $K = \mathsf{key}_1(a, B^*)$ | protocol description |
| 3 | $\mathsf{pw}^* \neq \perp \wedge \mathsf{pw} = \mathsf{pw}' \neq \mathsf{pw}^* \wedge (\phi')^* = \phi'$ | $K$ \$ | hybrid 4 |
| 4 | $\mathsf{pw} \neq \mathsf{pw}' \wedge (\phi')^* = \phi'$ | $K$ \$ | hybrid 3 |
| 5 | $\mathsf{pw} = \mathsf{pw}' \wedge \mathsf{pw}^* = \perp \wedge B^* = B \wedge$ $(\phi')^*$ contains the correct password guess (re-encryption case) | $K = K'$ | hybrid 1 |
| 6 | all other cases where $(\phi')^*$ contains the correct password guess | $K = \mathsf{key}_1(a, B^*)$ | protocol description |
| 7 | $(\phi')^*$ contains a wrong password guess | $K$ \$ | hybrid 3 |

Table 8: Definition of $K$ in hybrid 4. Both eavesdropping and re-encryption cases assume the two parties' passwords match

**Changing session keys.** Now we consider the outputs of $P$ and $P'$ under a variety of cases (note that up until now we have changed only KA keys $K, K'$, not the PAKE session keys that $P$ and $P'$ output). Let $SK$ be the session key of $P$ and $SK'$ be the session key of $P'$.

The following hybrids 5–7 are immediate (hybrid 5 is purely conceptual, while hybrids 6 and 7 uses the fact that $\mathsf{PRF}$ is a PRF):

*Hybrid 5:* In case 1 of Table 8, set $SK := SK'$.

*Hybrid 6:* In cases 1, 2 and 4 of Table 7, replace $\mathsf{PRF}_{K'}$ with a random function $G'$.

*Hybrid 7:* In cases 3, 4 and 7 of Table 8, replace $\mathsf{PRF}_K$ with a random function $G$.

The following hybrids 8 and 9 deal with case 5 (re-encryption).

*Hybrid 8:* In case 5 of Table 8, set $SK := \mathsf{PRF}_K((\phi')^*)$.

This case is covered by hybrid 1, which sets $K := K'$ (and thus $SK = \mathsf{PRF}_K((\phi')^*) = \mathsf{PRF}_{K'}((\phi')^*)$); then by hybrid 6, which changes $(SK, SK')$ from $(\mathsf{PRF}_{K'}((\phi')^*), \mathsf{PRF}_{K'}(\phi'))$ to $(G'((\phi')^*), G'(\phi'))$. Since $(\phi')^* \neq \phi'$, $SK$ and $SK'$ are independent of each other, so changing $SK$ back to $\mathsf{PRF}_K((\phi')^*)$ generates an indistinguishable view.

At this point, in the cases when $K = K'$ (eavesdropping and re-encryption), $P$ does not use a random function to output (hybrids 5 and 8 define $SK$ specifically in these cases). Therefore $P$ and $P'$ don't ever use the same $G$ or $G'$ to output: $P'$ outputs with $G'$ and $P$ outputs with $G$, with $G, G'$ independently drawn.

*Hybrid 9:* In case 5 of Table 8, $P$ sets $K := \mathsf{key}_1(a, B)$.

Note that in hybrid 8 $K = K'$ is only used while computing $SK$ (in particular, the $P'$ side has been changed to all random and does not use $K'$). In hybrid 8 $K \leftarrow \mathcal{K}$, whereas in hybrid 9 $K = \mathsf{key}_1(a, B)$ for $A = \mathsf{msg}_1(a)$ and $B \leftarrow \mathcal{M}_2$. Thus, a reduction to Lem. 5.5 (that uses its challenge $K$ to compute $SK$ for $P$) shows that hybrids 8 and 9 are indistinguishable.

**Comparison between hybrid 9 and the ideal world.** We now argue that hybrid 9 is identical to the ideal world, where the simulator $\mathcal{S}$ as defined in Figure 20 interacts with $\mathcal{F}_{\text{PAKE}}$. We first consider the $P'$ side output $SK'$. We refer to cases in Table 7:

- In cases 1 and 2, in hybrid 9 $SK'$ is uniformly random. In the ideal world, $\mathcal{S}$ enters step 8, where it sends NewKey without TestPwd, so the $P'$ session is fresh and $\mathcal{F}_{\text{PAKE}}$ samples a uniformly random $SK'$ for $P'$.

- In case 3, in hybrid 9 $SK' = \mathsf{PRF}_{K'}(\phi')$ where $K' = \mathsf{key}_2(b, A^*)$. In the ideal world, $\mathcal{S}$ enters step 9, where it sends TestPwd resulting in "correct guess" and then NewKey where the session key is $\mathsf{PRF}_{K'}(\phi')$ where $K' = \mathsf{key}_2(b, A^*)$, so the $P'$ session is compromised and $\mathcal{F}_{\text{PAKE}}$ sets $SK' := \mathsf{PRF}_{K'}(\phi')$ for $P'$.

- In case 4, in hybrid 9 $SK'$ is uniformly random. In the ideal world, $\mathcal{S}$ enters step 9, where it sends TestPwd resulting in "wrong guess" and then NewKey, so the $P'$ session is interrupted and $\mathcal{F}_{\text{PAKE}}$ samples a uniformly random $SK'$ for $P'$. (Note that in this case the session key chosen by $\mathcal{S}$ in NewKey does not matter.)

We conclude that for $SK'$, hybrid 9 and the ideal world are identical in all cases.

Next, we consider the $P$ side output $SK$. We refer to cases in Table 8:

- In case 1 (eavesdropping), in hybrid 9 $SK = SK'$. In the ideal world, $\mathcal{S}$ enters step 10(1), where it sends NewKey without TestPwd, so the $P$ session is fresh. Since the $P'$ session was also fresh when $P'$ output and $\mathsf{pw} = \mathsf{pw}'$, $\mathcal{F}_{\text{PAKE}}$ sets $SK := SK'$ for $P$.

- In case 2, in hybrid 9 $SK = \mathsf{PRF}_K((\phi')^*) = \mathsf{PRF}_K(\phi')$ where $K = \mathsf{key}_1(a, B^*)$. Note that $B^* = F_{(\phi')^*}(\mathsf{pw}) = F_{\phi'}(\mathsf{pw}') = B = \mathsf{msg}_2(b, A^*)$. In the ideal world, $\mathcal{S}$ enters step 11, where it sends TestPwd resulting in "correct guess" and then NewKey where the session key is $\mathsf{PRF}_{K^*}(\phi')$ and $K^* = \mathsf{key}_1(a, \mathsf{msg}_2(b, A^*))$. Then the $P$ session is compromised and $\mathcal{F}_{\text{PAKE}}$ sets $SK := \mathsf{PRF}_{K^*}(\phi')$ for $P$. We can see that in both hybrid 9 and the ideal world, $SK = \mathsf{PRF}_{\mathsf{key}_1(a,\mathsf{msg}_2(b,A^*))}(\phi')$.

- In case 3, in hybrid 9 $SK$ is uniformly random. In the ideal world, $\mathcal{S}$ enters step 10(2), where it sends NewKey without TestPwd, so the $P$ session is fresh. Since the $P'$ session was interrupted by a wrong password guess when $P'$ output, $\mathcal{F}_{\text{PAKE}}$ samples a uniformly random $SK$ for $P$.

- In case 4, in hybrid 9 $SK$ is uniformly random. In the ideal world there are several sub-cases: (i) if $\mathsf{pw}^* = \perp$, then $\mathcal{S}$ enters step 10(1). This is identical to case 1, except that the $P'$ session was fresh when $P'$ output but $\mathsf{pw} \neq \mathsf{pw}'$, so $\mathcal{F}_{\text{PAKE}}$ samples a uniformly random $SK$ for $P$ (independent of $SK'$); (ii) if $\mathsf{pw}^* = \mathsf{pw}' \neq \mathsf{pw}'$, then $\mathcal{S}$ enters step 11, where it sends TestPwd resulting in "wrong guess" and then NewKey, so the $P$ session is interrupted and $\mathcal{F}_{\text{PAKE}}$ samples a uniformly random $SK$ for $P$; (iii) if $\mathsf{pw}^* \neq \perp \wedge \mathsf{pw}^* \neq \mathsf{pw}'$, then $\mathcal{S}$ enters step 10(2), so $SK$ is uniformly random by the same argument as in case 3.

- Case 5 (re-encryption) is the most complicated one because it is changed five times: in hybrids 1, 2, 6, 8 and 9, where both $K$ and $SK$ are changed back and forth between the "real" value and uniformly random. Eventually in hybrid 9 $SK = \mathsf{PRF}_K((\phi')^*)$ where $K = \mathsf{key}_1(a, B)$. In the ideal world, $\mathcal{S}$ enters step 12, where it sends TestPwd resulting in "correct guess" and then NewKey where the session key is $\mathsf{PRF}_{K^*}((\phi')^*)$ and $K^* = \mathsf{key}_1(a, (B')^*)$. Note that in this case $(B')^* = B^* = B$. Then the $P$ session is compromised and $\mathcal{F}_{\text{PAKE}}$ sets $SK := \mathsf{PRF}_{K^*}((\phi')^*)$ for $P$. We can see that in both hybrid 9 and the ideal world, $SK = \mathsf{PRF}_{\mathsf{key}_1(a,B)}((\phi')^*)$.

- Case 6 is similar to case 5, except that $B^*$ might not equal $B$. In hybrid 9 $SK = \mathsf{PRF}_K((\phi')^*)$ where $K = \mathsf{key}_1(a, B^*)$. Since $(\phi')^*$ contains a correct password guess, $(\mathsf{pw}')^* = \mathsf{pw}$, so $B^* = F_{(\phi')}(\mathsf{pw}) = F_{(\phi')}((\mathsf{pw}')^*) = (B')^*$. The ideal world is identical to case 5: $\mathcal{F}_{\text{PAKE}}$ sets

$SK := \mathsf{PRF}_{K^*}((\phi')^*)$ for $P$, where $K^* = \mathsf{key}_1(a, (B')^*)$. We can see that in both hybrid 9 and the ideal world, $SK = \mathsf{PRF}_{\mathsf{key}_1(a,(B')^*)}((\phi')^*)$.

- In case 7, in hybrid 9 $SK$ is uniformly random. In the ideal world, $\mathcal{S}$ enters step 12, where it sends TestPwd resulting in "wrong guess" and then NewKey, so the $P'$ session is interrupted and $\mathcal{F}_{\text{PAKE}}$ samples a uniformly random $SK$ for $P$.

We conclude that for $SK$, hybrid 9 and the ideal world are identical in all cases.

We also must argue that the simulation of $\mathcal{F}_{\text{POPF}}$ is indistinguishable between hybrid 9 and the ideal world. The outputs of the POPF are independent of the rest of the simulation, except for the programmed ones. $\mathcal{A}$ already knows any outputs it programs, but $P'$ also programs an output: $F_{\phi'}(\mathsf{pw}') = B$. Therefore if $\mathcal{A}$ ever guesses $\mathsf{pw}'$ correctly it will know $B$, and so we must adjust the simulation to output $B$ in this case — which is exactly what our simulator does. Note that it suffices to only simulate this *after* the password guess on Deliver, as $\mathcal{A}$ must provide a $\phi'$ that has never been evaluated previously; therefore $\mathcal{A}$ cannot evaluate $F_{\phi'}(\mathsf{pw}')$ before the simulator can program that output into $\mathcal{F}_{\text{POPF}}$.

Finally, the careful reader will note that our argument that hybrid 9 and the ideal world are identical assumes that $\mathcal{A}$ delivers all messages (possibly after modification). If $\mathcal{A}$ drops the message to one party or the other, that does not affect the indistinguishability of the simulation: since that party no longer outputs, we no longer need to simulate them. For example, the simulation of $\mathcal{A}$ modifying the second ($P'$-to-$P$) message is identical to the simulation of $\mathcal{A}$ dropping the first message and supplying its own second message to $P$ (as if $P'$ does not exist). Even if $\mathcal{A}$ chooses to send messages out of order this still does not affect simulation. For example, suppose when $\mathcal{A}$ receives Deliver, it first sends $(\phi')^*$ to $P$ so $P$ outputs, and then sends the Deliver message to $\mathcal{F}_{\text{PAKE-1r}}$. Since $\mathcal{A}$ does not see the output of $P$ after it sends $(\phi')^*$, without loss of generality we may assume that these messages are sent in the opposite order, reducing $\mathcal{A}$ to an adversary that delivers all messages (possibly after modification).

We conclude that hybrid 9 and the ideal world are identical in all cases. This completes the proof. □

The theorem immediately implies the following: let KA be a correct, secure, strongly pseudorandom, and pseudorandom non-malleable KA protocol. Beginning with the protocol in Figure 19, replace $\mathcal{F}_{\text{PAKE-1r}}$ with the protocol in Figure 17, and $\mathcal{F}_{\text{POPF}}$ with the protocol in Figure 11. Then by Lem. 5.1 and Thms. 4.2 and 5.4 the resulting protocol realizes $\mathcal{F}_{\text{PAKE}}$ in the ROM.

### 5.3 The OEKE Protocol

Parameters:
- EKE first round functionality $\mathcal{F}_{\text{PAKE-1r}}$ (see Figure 16) for a key agreement protocol $\mathsf{KA}$.

On input $(\mathsf{NewSession}, \mathsf{sid}, P, P', \mathsf{pw}, \text{“initiator”})$, party $P$ does the following:
1. Sample and record $a \leftarrow \mathcal{R}$, and compute $A := \mathsf{msg}_1(a)$.
2. Send $(\mathsf{Program}, sid, P, P', \mathsf{pw}, A)$ to $\mathcal{F}_{\text{PAKE-1r}}$.

On input $(\mathsf{NewSession}, \mathsf{sid}, P', P, \mathsf{pw}', \text{“respondent”})$, party $P'$ sends $(\mathsf{SampleResp}, sid, P, P', \mathsf{pw}')$ to $\mathcal{F}_{\text{PAKE-1r}}$.

On message $(\mathsf{sid}, B, K' \,\|\, \tau')$ from $\mathcal{F}_{\text{PAKE-1r}}$, party $P'$ does the following:
3. Send $(\mathsf{sid}, B, \tau')$ to $P$.
4. Output $(\mathsf{sid}, K')$.

On message $(\mathsf{sid}, B, \tau')$ from $P'$, party $P$ does the following:
5. Retrieve $a$ and compute $K \,\|\, \tau := \mathsf{key}_1(a, B)$.
6. Check if $\tau = \tau'$. If so, output $(\mathsf{sid}, K)$. Otherwise output $(\mathsf{sid}, K_\$)$ where $K_\$ \leftarrow \{0,1\}^\kappa$.

Figure 21: OEKE protocol, in the $\mathcal{F}_{\text{PAKE-1r}}$-hybrid model.

**Theorem 5.6.** *Suppose that* $\mathsf{KA}$ *is a correct, pseudorandom non-malleable, and collision resistant KA protocol, whose key space* $\mathcal{K} = \{0,1\}^{3\kappa}$. *Then the OEKE protocol (Figure 21) realizes* $\mathcal{F}_{\text{PAKE}}$ *in the* $(\mathcal{F}_{\text{POPF}}, \mathcal{F}_{\text{PAKE-1r}})$*-hybrid world.*[24]

| # | case addressed | change | property used |
|---|---|---|---|
| 1 | $\mathsf{pw}^* = \bot \wedge \mathsf{pw} = \mathsf{pw}'$ $\wedge B^* = B$ | $K = K'$ if $\tau^* = \tau'$<br>$K$ \$ otherwise | KA correctness |
| 2 | $\mathsf{pw}^* = \bot \wedge \mathsf{pw} = \mathsf{pw}'$ | $K', \tau'$ \$ | KA pseudorandom non-malleability |
| 3 | $\mathsf{pw}^* = \bot \wedge \mathsf{pw} \neq \mathsf{pw}'$ $\wedge B^* = B \wedge \tau^* = \tau'$ | $P$ outputs $K_\$$ | none |
| 4 | $\mathsf{pw}^* \neq \bot$ $\vee B^* \neq B \vee \tau^* \neq \tau'$ | exclude collisions while extracting “good” $(\mathsf{pw}')^*$ | KA collision resistance |
| 5 | | $K = \mathsf{key}_1(a^*, B^*)[1]$ if $\mathsf{pw}$ “good”<br>$K$ \$ otherwise | KA collision resistance |
| 6 | | extract “good” $(\mathsf{pw}')^*$<br>$K = \mathsf{key}_1(a^*, B^*)[1]$ if $(\mathsf{pw}')^* = \mathsf{pw}$<br>$K$ \$ if $(\mathsf{pw}')^* \neq \mathsf{pw}$ or no $(\mathsf{pw}')^*$ | none |

Table 9: Summary of hybrids for OEKE security. “\$” denotes “chosen at random from respective range”.

*Proof.* The simulator $\mathcal{S}$ is shown in Figure 22. In this proof we use the following convention: for a $3\kappa$-bit string $s$, $s[1]$ denotes its first $\kappa$ bits (the $K$ part), and $s[2]$ denotes its last $2\kappa$ bits (the $\tau$ part).

The proof goes by the following hybrid argument (see Table 9 for a summary):

[24] Our protocol realizing $\mathcal{F}_{\text{PAKE-1r}}$ (Figure 17) additionally requires $\mathsf{KA}$ to be secure and pseudorandom.

Initialize $T := \{\}$ as the set of $\mathcal{F}_{\text{PAKE-1r}}$ evaluations.
On $(\mathsf{NewSession}, \mathsf{sid}, P, P', \text{“initiator”})$ from $\mathcal{F}_{\text{PAKE}}$:

1. Send $(\mathsf{Program}, \mathsf{sid}, P, P')$ from $\mathcal{F}_{\text{PAKE-1r}}$ to $\mathcal{A}$.

On $(\mathsf{NewSession}, \mathsf{sid}, P', P, \text{“respondent”})$ from $\mathcal{F}_{\text{PAKE}}$:

2. Send $(\mathsf{SampleResp}, \mathsf{sid}, P', P)$ from $\mathcal{F}_{\text{PAKE-1r}}$ to $\mathcal{A}$.

On $(\mathsf{Eval}, \mathsf{sid}, P, P', x)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{PAKE-1r}}$:

3. If there exists $(x, A, a) \in T$, return $A$ to $\mathcal{A}$.
4. Otherwise, sample $a \leftarrow \mathcal{R}$ and $A := \mathsf{msg}_1(a)$.
5. Add $(x, A, a)$ to $T$, and return $A$ to $\mathcal{A}$.

On $(\mathsf{Deliver}, \mathsf{sid}, P, P', \mathsf{pw}^*, A^*)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{PAKE-1r}}$:

6. If $\mathsf{pw}^* = \bot$, send $(\mathsf{NewKey}, \mathsf{sid}, P', 0^\kappa)$ to $\mathcal{F}_{\text{PAKE}}$. Furthermore, sample $B \leftarrow \mathcal{M}_2$ and $\tau' \leftarrow \{0,1\}^{2\kappa}$, and send $(\mathsf{sid}, B, \tau')$ from $P'$ to $P$.
7. Otherwise do the following steps:
   (1) Send $(\mathsf{TestPwd}, \mathsf{sid}, P', \mathsf{pw}^*)$ to $\mathcal{F}_{\text{PAKE}}$, and procede according to its response:

   **“correct guess”:** Sample $b \leftarrow \mathcal{R}$ and compute $B := \mathsf{msg}_2(b, A^*)$ and $K' \parallel \tau' := \mathsf{key}_2(b, A^*)$.

   **“wrong guess”:** Sample $(B, K', \tau') \leftarrow \mathcal{M}_2 \times \{0,1\}^\kappa \times \{0,1\}^{2\kappa}$.

   (2) Send $(\mathsf{NewKey}, \mathsf{sid}, P', K')$ to $\mathcal{F}_{\text{PAKE}}$ and $(\mathsf{sid}, B, \tau')$ from $P'$ to $P$.

On $(\mathsf{sid}, B^*, \tau^*)$ from $\mathcal{A}$ to $P$:

8. If $\mathsf{pw}^* = \bot \wedge B^* = B \wedge \tau^* = \tau'$, send $(\mathsf{NewKey}, \mathsf{sid}, P, 0^\kappa)$ to $\mathcal{F}_{\text{PAKE}}$.
9. Otherwise (i.e., if $\mathsf{pw}^*$ is undefined because no $\mathsf{Deliver}$ message has been sent, $\mathsf{pw}^* \neq \bot$, $B^* \neq B$, or $\tau^* \neq \tau'$), then for every $(x, A, a) \in T$, compute $K \parallel \tau = \mathsf{key}_1(a, B^*)$ and check whether $\tau^* = \tau$.
   A. If there is more than one such entry, output $\mathsf{Collision}$ and abort.
   B. If there is exactly one such entry, send $(\mathsf{TestPwd}, \mathsf{sid}, P, x)$ and then $(\mathsf{NewKey}, \mathsf{sid}, P, K)$ to $\mathcal{F}_{\text{PAKE}}$.
   C. If there is no such entry, send $(\mathsf{TestPwd}, \mathsf{sid}, P, \bot)$ and then $(\mathsf{NewKey}, \mathsf{sid}, P, 0^\kappa)$ to $\mathcal{F}_{\text{PAKE}}$.

Figure 22: Simulator $\mathcal{S}$ for the OEKE protocol.

*Hybrid 0:* This is the real world. Recall that the passwords of $P$ and $P'$ are $\mathsf{pw}$ and $\mathsf{pw}'$, respectively; the flow of events is:

- $P$ tells $\mathcal{F}_{\text{PAKE-1r}}$ to $\mathsf{Program}$ a function mapping $\mathsf{pw} \mapsto A$ and send it to $P'$;
- It is intercepted by the man-in-the-middle adversary $\mathcal{A}$, who possibly modifies it to program $\mathsf{pw}^* \mapsto A^*$ instead;
- $\mathcal{A}$ tells $\mathcal{F}_{\text{PAKE-1r}}$ to $\mathsf{Deliver}$ the function to $P'$;
- $P'$ tells $\mathcal{F}_{\text{PAKE-1r}}$ to $\mathsf{SampleResp}$, which generates $B, K', \tau'$;
- $P'$ outputs $K'$ and sends $(B, \tau')$ to $P$ (again intercepted by $\mathcal{A}$);
- $\mathcal{A}$ sends $(B^*, \tau^*)$ to $P$;
- $P$ outputs $K$ if $\tau^*$ matches $\tau$, and a random $K_\$$ otherwise.

**The case that $\mathsf{pw}^* = \perp$.** The following hybrids 1–3 only affect the case where $\mathsf{pw}^* = \perp$, i.e., $\mathcal{A}$ does not modify the $P$-to-$P'$ message.

*Hybrid 1:* In the case that $\mathsf{pw}^* = \perp \wedge \mathsf{pw} = \mathsf{pw}' \wedge B^* = B$, do the following:

- If $\tau^* = \tau'$ (i.e., $\mathcal{A}$ is an eavesdropper), then $P$ outputs $K := K'$;
- Otherwise, $P$ outputs $K_\$ \leftarrow \{0,1\}^\kappa$.

In hybrid 0, we have that

- If $\tau^* = \tau = \mathsf{key}_1(a, B^*)[2]$, then $P$ outputs $K := \mathsf{key}_1(a, B^*)[1]$;
- Otherwise, $P$ outputs $K_\$ \leftarrow \{0,1\}^\kappa$.

The difference is that in hybrid 0, $P$ outputs $K$ if $\tau^* = \tau$, whereas in hybrid 1, $P$ outputs $K'$ (and defines $K$ as $K'$) if $\tau^* = \tau'$. Since $B^* = B$, we have $K \parallel \tau = \mathsf{key}_1(a, B^*) = \mathsf{key}_1(a, B)$; by the syntax of $\mathcal{F}_{\text{PAKE-1r}}$ in the case of $\mathsf{pw}^* = \perp$ and $\mathsf{pw} = \mathsf{pw}'$, $B = \mathsf{msg}_2(b, A)$ and $K' \parallel \tau' = \mathsf{key}_2(b, A)$, where $A = \mathsf{msg}_1(a)$ according to protocol description (step 1). Then by correctness of $\mathsf{KA}$, $(K, \tau) = (K', \tau')$ with overwhelming probability, so hybrids 0 and 1 are (statistically) indistinguishable.

*Hybrid 2:* In the case that $\mathsf{pw}^* = \perp \wedge \mathsf{pw} = \mathsf{pw}'$, sample $B \leftarrow \mathcal{M}_2$ and $K' \parallel \tau' \leftarrow \{0,1\}^{3\kappa}$.

The difference between hybrids 1 and 2 is that in hybrid 1, $B = \mathsf{msg}_2(b, A)$ and $K' \parallel \tau' = \mathsf{key}_2(b, A)$, whereas in hybrid 2, $B \leftarrow \mathcal{M}_2$ and $K' \parallel \tau' \leftarrow \{0,1\}^{3\kappa}$. We construct a reduction $\mathcal{R}$ to the pseudorandom non-malleability of the underlying key agreement protocol $\mathsf{KA}$: $\mathcal{R}$, given $(A, B, K', \tau')$, can simulate the experiment up to when $P'$ outputs — which includes $\mathcal{A}$'s access to $\mathcal{F}_{\text{PAKE-1r}}$, as well as $B$, $K'$ and $\tau'$ — without knowing $a$ or $b$. (Note that in $\mathcal{F}_{\text{PAKE-1r}}$, $b$ is used only when computing $B$ and $K'$.) When $\mathcal{A}$ sends $(B^*, \tau^*)$, $\mathcal{R}$ checks whether $B^* \neq B$. If $B^* \neq B$, $\mathcal{R}$ outputs $B^*$ to the pseudorandom non-malleability challenger and receives $K \parallel \tau$, then checks if $\tau^* = \tau$ and accordingly outputs either $K$ or a random string to $\mathcal{Z}$. Alternatively, if $B^* = B$, the pseudorandom non-malleability experiment prohibits $\mathcal{R}$ from outputting $B^*$. Instead, $\mathcal{R}$ picks some other element of $\mathcal{M}_2$ to output (just to finish the experiment), and uses $K' \parallel \tau'$[25] instead of $K \parallel \tau$ to choose what to output to $\mathcal{Z}$. $\mathcal{R}$ copies $\mathcal{Z}$'s output bit. We can see that if $B = \mathsf{msg}_2(b, A)$ and $K' \parallel \tau' = \mathsf{key}_2(b, A')$ then $\mathcal{R}$ simulates hybrid 1, and if $B \leftarrow \mathcal{M}_2$ and $K' \parallel \tau' \leftarrow \{0,1\}^{3\kappa}$ then $\mathcal{R}$ simulates hybrid 2. Thus,

[25] In hybrid 1 we already changed $P$ to use $K' \parallel \tau' = \mathsf{key}_2(b, A)$ instead of $\mathsf{key}_1(a, B)$ in this case.

$\mathcal{R}$'s distinguishing advantage is equal to $\mathcal{Z}$'s distinguishing advantage between hybrids 1 and 2, so hybrids 1 and 2 are indistinguishable.

*Hybrid 3:* In the case that $\mathsf{pw} \neq \mathsf{pw}' \wedge \mathsf{pw}^* = \bot \wedge B^* = B \wedge \tau^* = \tau'$, $P$ outputs $K_\$ \leftarrow \{0,1\}^\kappa$ (i.e., the check of $P$ always fails).

The difference between hybrids 2 and 3 is that in hybrid 2, $P$ computes $K \parallel \tau = \mathsf{key}_1(a, B^*) = \mathsf{key}_1(a, B)$ and outputs $K_\$ \leftarrow \{0,1\}^\kappa$ only when $\tau \neq \tau^* = \tau'$, whereas in hybrid 3, $P$ always outputs $K_\$ \leftarrow \{0,1\}^\kappa$. Note that $\mathcal{F}_{\text{PAKE-1r}}$ outputs a uniformly random $\tau' \leftarrow \{0,1\}^{2\kappa}$, so the probability that $\tau = \tau'$ is negligible. Thus, hybrids 2 and 3 are indistinguishable.

**All cases where the adversary is not an eavesdropper.** Both the case where $\mathsf{pw}^* \neq \bot$ (i.e., $\mathcal{A}$ modifies the $P$-to-$P'$ message, or chooses its own message first), and the case where $\mathsf{pw}^* = \bot$ and $B^* \neq B \vee \tau^* \neq \tau'$ (i.e., $\mathcal{A}$ modifies the $P'$-to-$P$ message) are affected by the following hybrids 4–6.

*Hybrid 4:* In the case that $\mathsf{pw}^* \neq \bot \vee B^* \neq B \vee \tau^* \neq \tau'$, output $\mathsf{Collision}$ and abort if there exists more than one "good" $(\mathsf{pw}')^*$. Here, and in subsequent hybrids, we call $(\mathsf{pw}')^*$ "good" if (1) there was a query $(\mathsf{Eval}, \mathsf{sid}, P, P', (\mathsf{pw}')^*)$ to $\mathcal{F}_{\text{PAKE-1r}}$ and (2) $\tau^* = \mathsf{key}_1(a^*, B^*)[2]$. (Intuitively a "good" $(\mathsf{pw}')^*$ is a password guess for $P$ that can be extracted from $(B^*, \tau^*)$.)

We upper-bound $\Pr[\mathsf{Collision}]$ via a reduction to the collision-resistance of $\mathsf{KA}$. Suppose that $\mathcal{F}_{\text{PAKE-1r}}$ receives at most $q$ distinct $\mathsf{Eval}$ queries. The reduction, given $(A_1, \ldots, A_q)$, handles the $i$-th such query by outputting $A_i$, without knowing the corresponding "trapdoor" $a_i$. Then the reduction simulates the $P'$ side to $\mathcal{Z}$, producing some $B$, $K'$ and $\tau'$. (Note that after hybrid 2, the only case where these are not sampled uniformly at random is that $\mathsf{pw}^* = \mathsf{pw}'$. In this case $B = \mathsf{msg}_2(b, A^*)$ and $K' \parallel \tau' = \mathsf{key}_2(b, A^*)$, where $A^*$ is chosen by $\mathcal{A}$, so the simulation of this case also does not require knowledge of $a$.) When $\mathcal{A}$ sends $(B^*, \tau^*)$, the reduction outputs $B^*$. If $\mathsf{Collision}$ happens, then the reduction wins. By the collision-resistance of $\mathsf{KA}$, $\Pr[\mathsf{Collision}]$ is negligible and thus hybrids 3 and 4 are indistinguishable.

*Hybrid 5:* In the case that $\mathsf{pw}^* \neq \bot \vee B^* \neq B \vee \tau^* \neq \tau'$, $P$ outputs $K_\$ \leftarrow \{0,1\}^\kappa$ if $\mathsf{pw}$ is not "good".

The difference between hybrids 4 and 5 occurs when $\tau^* = \tau$ (i.e., the check of $P$ passes), but $\mathsf{pw}$ is never sent to $\mathcal{F}_{\text{PAKE-1r}}$ via $\mathsf{Eval}$ (otherwise $\mathsf{pw}$ would be "good"). Then in hybrid 4, $P$ outputs $K = \mathsf{key}_1(a, B^*)[1]$, except for the special case where $\mathsf{pw}^* = \bot \wedge \mathsf{pw} = \mathsf{pw}' \wedge B^* = B$, which already outputs a random $K_\$$ because of hybrid 1. In hybrid 5, $P$ instead outputs $K_\$$. We argue that the probability $\mathsf{pw}$ is never sent in an $\mathsf{Eval}$ query and $\tau^* = \tau$ is negligible, via a reduction to the collision-resistance of $\mathsf{KA}$.

Let $q \geq 2$ be a parameter to be decided later. The reduction ignores its input $(A_1, \ldots, A_q)$, and simulates the $P'$ side as in the reduction under hybrid 4. When $\mathcal{A}$ sends $B^*$, the reduction also outputs $B^*$.

Assuming $\mathsf{Eval}(\ldots, \mathsf{pw})$ is never queried, the correct $A$ $\mathsf{Programed}$ by $P$ is never used and is unknown to $\mathcal{A}$, so for any $i \in [q]$ we could pretend that $A = A_i$ and it would make no difference to the probability that $\tau^* = \tau$. Therefore, for any $i$, the probability $p$ (given that $B^*$ takes its value) that $\tau^* = \mathsf{key}_1(a, B^*)[2]$ is the same as the probability that $\tau^* = \mathsf{key}_1(a_i, B^*)[2]$. Then the chance of a collision is at least the chance that there exists $i \neq j$ such that $\tau^* = \mathsf{key}_1(a_i, B^*)[2] = \mathsf{key}_1(a_j, B^*)[2]$. These are just $q$ independent events, so this probability is exactly

$$p' = \sum_{k=2}^{q} \binom{q}{k} p^k (1-p)^{q-k} \geq \binom{q}{2} p^2 (1-p)^{q-2},$$

which is obviously increasing in both $p$ and $q$, so for a lower bound on $p'$ we can reduce $p$ to be at

most $\frac{1}{2}$, and then reduce $q$ so that $q \leq 2 + \frac{1}{2p}$. Then $(1-p)^{q-2} \geq (1-p)^{\frac{1}{2p}} = 2^{\frac{\log_2(1-p)}{2p}} \geq \frac{1}{2}$, since $\log_2(1-p) \geq -2p$ for $p \leq \frac{1}{2}$. Substituting $q \mapsto \min(q, 2 + \frac{1}{2p})$ into $p'$ gives

$$p' \geq \frac{1}{4}\left(\min\left(q, 2+\frac{1}{2p}\right) - 1\right)^2 p^2 = \frac{1}{4}\min\left((q-1)^2p^2, \left(2+\frac{1}{2p}-1\right)^2 p^2\right) \geq \frac{1}{4}\min\left((q-1)^2p^2, \frac{1}{4}\right),$$

which is valid for $p \leq \frac{1}{2}$. Next, to make it always valid, substitute $p \mapsto \min(p, \frac{1}{2})$ to get

$$p' \geq \frac{1}{4}\min\left((q-1)^2 \min\left(p, \frac{1}{2}\right)^2, \frac{1}{4}\right) \geq \frac{1}{4}\min\left((q-1)^2p^2, \frac{1}{4}\right),$$

so this inequality is actually always valid.

To summarize, we have related $p$ (the probability of the "bad event" that allows $\mathcal{Z}$ to distinguish between hybrids 4 and 5) and $p'$ (a lower bound of the reduction's advantage against the collision-resistance of $\mathsf{KA}$). Now for any polynomial $q$, if the reduction has advantage $\mathsf{Adv}_q = p' < \frac{1}{16}$, then by the inequality above, $\mathsf{Adv}_q \geq \frac{1}{4}(q-1)^2p^2$. Therefore, hybrids 4 and 5 can be distinguished with probability at most

$$p < \frac{2}{q-1}\sqrt{\mathsf{Adv}_q}.$$

*Hybrid 6:* In the case that $\mathsf{pw}^* \neq \bot \vee B^* \neq B \vee \tau^* \neq \tau'$, do:

- If there is exactly one "good" $(\mathsf{pw}')^*$, and $(\mathsf{pw}')^* = \mathsf{pw}$, then $P$ outputs $K := \mathsf{key}_1(a, B^*)[1]$;
- If there is exactly one "good" $(\mathsf{pw}')^*$, and $(\mathsf{pw}')^* \neq \mathsf{pw}$, then $P$ outputs $K_\$ \leftarrow \{0,1\}^\kappa$;
- If there is no "good" $(\mathsf{pw}')^*$, then $P$ also outputs $K_\$ \leftarrow \{0,1\}^\kappa$.

The difference between hybrids 5 and 6 is that

- In hybrid 5, $P$ outputs $K$ if $\mathsf{pw}$ is "good", and $K_\$$ otherwise;
- In hybrid 6, $P$ outputs $K$ if there exists exactly one "good" $(\mathsf{pw}')^*$ and $(\mathsf{pw}')^* = \mathsf{pw}$, and $K_\$$ otherwise.

(If there is more than one "good" $(\mathsf{pw}')^*$, then both hybrids 5 and 6 output $\mathsf{Collision}$ and abort.)

Assuming $\mathsf{Collision}$ does not happen, there is *at most* one "good" $\mathsf{pw}^*$. If $\mathsf{pw}$ is "good", this means that there is exactly one "good" $\mathsf{pw}^*$ and $\mathsf{pw}^* = \mathsf{pw}$. Vice versa, if there is exactly one "good" $\mathsf{pw}^*$ and $\mathsf{pw}^* = \mathsf{pw}$, then of course $\mathsf{pw}$ is "good". Thus, the conditions in hybrids 5 and 6 are equivalent, so hybrids 5 and 6 are identical.

**Comparison between hybrid 6 and the ideal world.** We now argue that hybrid 6 is identical to the ideal world, where the simulator $\mathcal{S}$ as defined in Figure 22 interacts with $\mathcal{F}_{\text{PAKE}}$. We first consider the $P'$ side message $B, \tau'$ and session key $K'$. This is very similar to the corresponding argument in the proof of Thm. 5.4: in hybrid 6 (in fact, after hybrid 2) the only case where $B$, $\tau'$ and $K'$ are not uniformly random is that $\mathsf{pw}^* = \mathsf{pw}'$; in this case $B = \mathsf{msg}_2(b, A^*)$ and $K' \,\|\, \tau' = \mathsf{key}_2(b, A^*)$ (cf. Table 7). In the ideal world,

- If $\mathsf{pw}^* = \bot$, $\mathcal{S}$ enters step 6, where it sends $\mathsf{NewKey}$ without $\mathsf{TestPwd}$, so the $P'$ session is $\mathsf{fresh}$ and $\mathcal{F}_{\text{PAKE}}$ samples a uniformly random $K'$ for $P'$. $\mathcal{S}$ also samples uniformly random $B$ and $\tau'$ as the message.

- If $\mathsf{pw}^* = \mathsf{pw}'$, $\mathcal{S}$ enters step 7, where it sends TestPwd resulting in "correct guess" and then NewKey where the session key is $\mathsf{key}_2(b, A^*)[1]$ where $b \leftarrow \mathcal{R}$, so the $P'$ session is compromised and $\mathcal{F}_{\text{PAKE}}$ sets $K' := \mathsf{key}_2(b, A^*)[1]$ for $P'$. $\mathcal{S}$ also computes $B = \mathsf{msg}_2(b, A^*)$ and $\tau' = \mathsf{key}_2(b, A^*)[2]$.

- If $\mathsf{pw}^* \neq \bot \wedge \mathsf{pw}^* \neq \mathsf{pw}'$, $\mathcal{S}$ enters step 7, where it sends TestPwd resulting in "wrong guess" and then NewKey, so the $P'$ session is interrupted and $\mathcal{F}_{\text{PAKE}}$ samples a uniformly random $K'$ for $P'$. (Note that in this case the session key chosen by $\mathcal{S}$ in NewKey does not matter.) $\mathcal{S}$ also samples uniformly random $B$ and $\tau'$.

We conclude that for $B$, $\tau'$ and $K'$, hybrid 6 and the ideal world are identical in all cases.

Next, we consider the $P$ side output, which is either $K$ or a uniformly random $K_{\$}$. Since there are fewer hybrids than in the proof in Thm. 5.4, here we do not make a table of all cases and state them in the text instead:

- If $\mathsf{pw} = \mathsf{pw}' \wedge \mathsf{pw}^* = \bot \wedge B^* = B \wedge \tau^* = \tau$ ($\mathcal{A}$ eavesdrops and the two parties' passwords match), in hybrid 6 $P$ outputs $K = K'$ (changed in hybrid 1). In the ideal world, $\mathcal{S}$ enters step 8, where it sends NewKey without TestPwd, so the $P$ session is fresh. Since the $P'$ session was also fresh when $P'$ output and $\mathsf{pw} = \mathsf{pw}'$, $\mathcal{F}_{\text{PAKE}}$ sets $K := K'$ for $P$.

- If $\mathsf{pw} \neq \mathsf{pw}' \wedge \mathsf{pw}^* = \bot \wedge B^* = B \wedge \tau^* = \tau$ ($\mathcal{A}$ eavesdrops and the two parties' passwords do not match), in hybrid 6 $P$ outputs $K_{\$}$ (changed in hybrid 3). In the ideal world, this is identical to the previous case, except that the $P'$ session was fresh when $P'$ output but $\mathsf{pw} \neq \mathsf{pw}'$, so $\mathcal{F}_{\text{PAKE}}$ samples a uniformly random session key for $P$ (independent of $K'$).

- If $\mathcal{A}$ does not eavesdrop and there are multiple "good" $(\mathsf{pw}')^*$, hybrid 6 outputs Collision and aborts the experiment (changed in hybrid 4). In the ideal world, $\mathcal{S}$ enters step 9A, where it also outputs Collision and aborts.

- If $\mathcal{A}$ does not eavesdrop, there is exactly one "good" $(\mathsf{pw}')^*$, and $(\mathsf{pw}')^* = \mathsf{pw}$, in hybrid 6 $P$ outputs $P$ outputs $K = \mathsf{key}_1(a, B^*)[1]$ (changed in hybrid 6). In the ideal world, $\mathcal{S}$ enters step 9B, where it sends TestPwd resulting in "correct guess" and then NewKey where the session key is $\mathsf{key}_1(a, B^*)[1]$ and $a$ is the value sampled by $(\mathsf{Eval}, \ldots, \mathsf{pw})$. Then the $P$ session is compromised and $\mathcal{F}_{\text{PAKE}}$ sets $K := \mathsf{key}_1(a, B^*)[1]$ for $P$. We can see that in both hybrid 9 and the ideal world, $K = \mathsf{key}_1(a, B^*)[1]$.

- If $\mathcal{A}$ does not eavesdrop, there is exactly one "good" $(\mathsf{pw}')^*$, and $(\mathsf{pw}')^* \neq \mathsf{pw}$, in hybrid 6 $P$ outputs $K_{\$}$ (changed in hybrid 6). In the ideal world, $\mathcal{S}$ enters step 9B, where it sends TestPwd resulting in "wrong guess" and then NewKey, so the $P$ session is interrupted and $\mathcal{F}_{\text{PAKE}}$ samples a uniformly random session key for $P$.

- If $\mathcal{A}$ does not eavesdrop and there is no "good" $(\mathsf{pw}')^*$, in hybrid 6 $P$ outputs $K_{\$}$ (changed in hybrid 5). In the ideal world, $\mathcal{S}$ enters step 9C, where it sends TestPwd resulting in "wrong guess" (note that $\mathcal{S}$'s password guess is $\bot$) and then NewKey, so the $P$ session is interrupted and $\mathcal{F}_{\text{PAKE}}$ samples a uniformly random session key for $P$.

We conclude that for $K$, hybrid 6 and the ideal world are identical in all cases. In summary, hybrid 6 and the ideal world are identical. This completes the proof. □

The theorem immediately implies the following: let KA be a correct, secure, pseudorandom, pseudorandom non-malleable, and collision-resistant KA protocol. Beginning with the protocol

in Figure 21, replace $\mathcal{F}_{\text{PAKE-1r}}$ with the protocol in Figure 17, and $\mathcal{F}_{\text{POPF}}$ with the protocol in Figure 11. Then by Lem. 5.1 and Thms. 4.2 and 5.6 the resulting protocol realizes $\mathcal{F}_{\text{PAKE}}$ in the ROM.

**OEKE-PRF and OEKE-RO.** Since our Thm. 5.6 considers a general OEKE protocol with *any* KA protocol whose key length is $3\kappa$, this immediately covers OEKE-PRF as a special case, where the KA protocol is obtained via taking another KA protocol whose key length is $\kappa$ and applying a PRF to the key (on inputs such as 0, 1 and 2). OEKE-RO is in turn a special case of OEKE-PRF, where the PRF is an RO.

# 6 Conclusion and Future Work

Why are there so many recent works on the UC-security of (O)EKE, the very first PAKE protocol that has been around for over 30 years? Why are there so many subtleties in the analysis of this simple and seemingly innocuous protocol? In this final section, we offer some insights and personal perspectives.

## 6.1 Subtleties in the Security Model

The first source of complication lies in the security model. To begin with, any security notion of PAKE *must consider a man-in-the-middle adversary*; in other words, the parties cannot have authenticated channels between each other. This setting is different from the vast majority of works on multi-party computation, and as we have seen in Sects. 3 and 5, the most complicated and subtle case is that the man-in-the-middle adversary passes the first message without modification but then modifies the second — which does not have a correspondence in "normal" 2PC where one party is honest and the other is corrupt. Failure to consider such cases (e.g., [MRR20]) renders the security proof incomplete and might even result in incorrect security statements.

Next, the UC PAKE functionality is also non-trivial and somewhat difficult to understand. PAKE is a quintessential example of a cryptographic primitive whose security notion is easy to see intuitively but hard to define formally. At first glance, the security requirement is simply "the only feasible attack is online guessing". But a complete description of the two parties' output behaviors must consider a large number of cases, as each party has three possible states: no attack, successfully attacked, and unsuccessfully attacked (corresponding to the sessions being fresh, compromised, and interrupted, respectively); furthermore, the output of one party depends on not only the state of itself, but also (if the party is unattacked) the state of its counterparty and whether the two parties' passwords match. This is why the UC PAKE functionality (Figure 3) contains some complicated sentences such as

> *If the record* $[\langle P, P', \mathsf{pw}\rangle]$ *is* fresh*, a key* $(\mathsf{sid}, K')$ *has been output to* $P'$*, at which time there was a record* $\langle P', P, \mathsf{pw}\rangle$ *marked* fresh*, then set* $K := K'$*.*

which essentially just says that if there is no attack on either session and the two parties' passwords match, then they should output the same (uniformly random) key. The complication here is that to formally describe this, the party that outputs first needs to output a random key, and the party that outputs next needs to output a key which is the same as the first party's. (It is interesting that this simplest case needs the most complicated language to describe.)

What complicates things even further is that *in UC, the order of events matters.* For example, consider a 2-round PAKE and assume the two parties' passwords match; if one direction is successfully

attacked and the other direction is not, one might expect that the unattacked party should output an independent random key. However, this is not necessarily true. Consider two cases: (1) the adversary passes the first ($P$-to-$P'$) message without modification, but then modifies the second ($P'$-to-$P$) and successfully attacks the $P$ session, and (2) the adversary modifies the first message and successfully attacks the $P'$ session, but then passes the second message. As we have seen in Sect. 3, in case (1) the unattacked $P'$ should indeed output an independent random key, as its session is fresh and *the simulator does not know the key of* $P'$. However, in case (2) when $P'$ outputs the simulator already knows *both the password and the session key of* $P'$, so even if the adversary does not modify the $P'$-to-$P$ session, the simulator can still use the password to compromise the $P$ session and set the session key of $P$ to be correlated to that of $P'$. In other words, the simulator in case (2) has more options and is "stronger". This explains why almost all flaws in prior works are failures to consider case (1), rather than case (2).

### 6.2 Subtleties in the Protocol Description

The second type of complexity comes from the fact that both EKE and OEKE have a large number of variants. These variants have two dimensions: whether IC, HIC or POPF is used in protocol messages, and how exactly the parties' session keys are derived. Since we have already poured a lot of ink on POPF, here we mainly focus on output derivation.

**The case of EKE.** In EKE there is, of course, the option to output the "raw" KA key $K$. However, it appears generally understood that the KA key needs to be hashed. Still, there are a lot of confusions on what exactly should be included in the hash: Should the transcript $\phi, \phi'$ be included? What about the password pw? And is it possible to avoid explicitly working in the ROM by using a PRF instead?

Indeed, it is highly non-trivial to see what needs to be hashed and what does not, and what exactly will go wrong if we don't hash those items that are necessary.[26] Now equipped with our (in)security results, we can give a summary here:

- Outputting the "raw" $K$ requires pseudorandom non-malleability of the underlying KA protocol, which covers an adversary that passes the first message but modifies the second (Sect. 3.1 and Appx. B).
- Outputting an RO hash of $K$ — which can be viewed as using a modified KA protocol whose key is $H(K)$ instead of $K$, then outputting the "raw" key of this modified KA protocol — essentially creates a new KA protocol that has pseudorandom non-malleability *in the specific case of KA protocols with perfect pseudorandomness (such as Diffie-Hellman)*; however, there is a $\Theta(q)$ loss while reducing to KA security (Sect. 3.3, the "reducing to DDH" case). Furthermore, if we take a secure and (strong) pseudorandom KA protocol and hash the key at the end, this does not yield a pseudorandom non-malleable KA protocol in general, so pseudorandom non-malleability is still needed (Appx. B).
- Including pw in the hash does not help, as all attacks on the "raw" protocol require the adversary to know pw (and query the IC accordingly), so the simulator can already extract pw from protocol messages and additionally allowing extraction from the final hash does not provide any advantage.

[26] Of course, the "safest" option is to simply hash everything. But this comes with the risk of writing a flawed security proof, as one would then be sure that this version "works". To put it in another way, given a complete security proof, it should be immediately clear what exact items have to be hashed.

- Including $\phi$ in the hash does not help either, as all attacks on the "raw" protocol require the adversary to pass $\phi$ without modification — in other words, the first message is consistent among (and known to) all parties including the adversary, so there is no need to hash it.
- Including $\phi'$ in the hash helps only when HIC or POPF is used, and the problematic case is again the adversary passes the first message but modifies the second. This time the adversary can replace $\phi'$ with another $(\phi')^*$ which corresponds to the same underlying KA message, and (standard) UC PAKE security dictates that in this case the two parties must output independent keys (Sect. 3.5) — which is achieved exactly by letting $P'$ output $H(\phi', K)$ and $P$ output $H((\phi')^*, K)$.
- Finally, a trivial observation is that $H(\phi', K)$ in the bullet above can be replaced by $\mathsf{PRF}_K(\phi')$, avoiding explicit mentioning of an RO. This is what our Thm. 5.4 analyzes.

Independent of the above, *strong* pseudorandomness of the underlying KA protocol is required no matter what is included in the hash (Appx. A).

See Table 10 for a summary of the discussion above.

| output function | insecure with plain Diffie-Hellman? | $\Theta(q^2)$ loss under CDH / $\Theta(q)$ loss under DDH with plain Diffie-Hellman? | only realizes $\mathcal{F}_{\text{PAKE-sp}}$ if HIC/POPF used? | analyzed in... |
|---|---|---|---|---|
| $K$ | ✓ | | ✓ | [MRR20, Theorem 10] [SGJ23, Theorem 2] (both theorems overlook both issues) our Thms. D.1 and D.2 |
| $H(K)$ | | ✓ | ✓ | |
| $H(\phi, \phi', K)$ | | ✓ | | [DHP+18, Theorem 6] [BCP+23, Theorem 1] (both proofs overlook the security loss) |
| $H(\mathsf{pw}, \phi, \phi', K)$ | | ✓ | | |
| $H(\phi', K)$ | | ✓ | | |
| $\mathsf{PRF}_K(\phi')$ (EKE-PRF) | ✓ | | | our Thm. 5.4 |
| In all cases, strong pseudorandomness and pseudorandom non-malleability needed in general (overlooked in [MRR20, Theorem 10], [SGJ23, Theorem 2], [BCP+23, Theorem 1]) | | | | |

Table 10: Summary of various versions of EKE. [DHP+18, Theorem 6] uses Diffie-Hellman KA and IC; [MRR20, Theorem 10] uses general KA and POPF; [SGJ23, Theorem 2] uses general KA and HIC; [BCP+23, Theorem 1] uses general KA and IC; our Thm. 5.4 uses general KA and POPF

**The case of OEKE.** The "raw" version of OEKE needs a KA protocol whose key is longer than the PAKE session key ($\kappa$ bits), as there are two things that need to appear independent of each other: the session key $SK$ and the authenticator $\tau$. Our result assumes such a KA protocol, but

existing works assume a KA protocol whose key is $\kappa$-bit long and then use specific methods to compile it into a KA protocol with long key:

- In OEKE-PRF, $SK = \mathsf{PRF}_K(0)$ and $\tau = \mathsf{PRF}_K(1)$ (where $F$ is a PRF). This version requires the underlying KA to be pseudorandom non-malleable, as an attack similar to that of EKE with "raw" key — where the adversary passes the first message and modifies the second — can cause the two parties' KA keys to be correlated, and the PRF offers no security guarantee in this case (Sect. 3.2). (A difference with the EKE attack is that here the adversary does not even need to know the password, as the second message is not encrypted.) A second attack involves the adversary (that also doesn't need to know the password) unilaterally biasing the KA key of $P$ and predicting the session key of $P$, revealing the necessity of collision resistance in the underlying KA protocol (Sect. 3.4).
- In OEKE-RO, $SK = H(K, 0)$ and $\tau = H(K, 1)$. Now the RO guarantees independent outputs even if the KA keys are correlated. However, a more sophisticated attack shows that pseudorandom non-malleability is still needed (Appx. B). Furthermore, this does not alleviate the second attack above, so collision resistance is also needed.
- Including the password $\mathsf{pw}$ in the hash *for* $\tau$ eliminates the second attack above, as coming up with a valid $\tau$ requires knowledge of $\mathsf{pw}$, and if the adversary knows $\mathsf{pw}$ (and uses it to attack the $P$ session), then we have to allow it to predict the session key of $P$ anyway. Therefore, collision resistance is not needed anymore (Remark 3.10). Note that including $\mathsf{pw}$ in the hash for the session key $SK$ does not provide any additional advantage.
- The transcript $\phi, (B, \tau)$ is not used in any of the attacks, so there is no need to including it in the hash.

See Table 11 for a summary of the discussion above.

| output function | collision resistance needed? | analyzed in... |
|---|---|---|
| $(\mathsf{PRF}_K(0), \mathsf{PRF}_K(1))$ (OEKE-PRF) | ✓ | [SGJ23, Theorem 3] proof (overlooks this property) |
| $(H(K,0), H(K,1))$ (OEKE-RO) | ✓ | [SGJ23, Theorem 3] statement (overlooks this property) |
| $(H(K), H(\mathsf{pw}, K))$ | | |
| $(H(\phi, B, \tau, K),$ $H(\mathsf{pw}, \phi, B, K))$ | | [BCP+23, Theorem 2] |
| $(K[1], K[2])$ (OEKE) | ✓ | our Thm. 5.6 |
| In all cases, pseudorandom non-malleability needed (overlooked in [SGJ23, Theorem 3], [BCP+23, Theorem 2]) | | |

Table 11: Summary of various versions of OEKE. [SGJ23, Theorem 3] uses general KA and HIC; [BCP+23, Theorem 2] uses general KA and IC; our Thm. 5.6 uses general KA and POPF

**Remark 6.1.** *For an example of whether hashing the password or not* while deriving the session key *actually matters, see [AP05] which proposes two PAKE protocols, SPAKE1 and SPAKE2, whose*

*only difference is that SPAKE2 includes the password in the final hash while SPAKE1 does not. Careful analysis shows that the (game-based) security of SPAKE1 is in the non-concurrent setting and non-tight under CDH, while SPAKE2 has concurrent security and a tight reduction to CDH.*

### 6.3 Subtleties in the Security Analysis

Regarding the security analysis, the general lesson is that in the context of UC, a reduction generally needs to act as the simulator (plus the functionality) while communicating with the environment/adversary; however, while performing the simulator's task, *the reduction loses some information compared with the actual simulator because it needs to embed some challenges in the experiment.* (Of course, "losing some information" is the case for reductions in general. But the complexity of the UC framework leads to this principle being overlooked more frequently.) As we have seen in Sect. 3.3, the flaws in the proofs of [BCP+23, Theorem 1] and [DHP+18, Theorem 6] are that in both proofs the reduction to CDH/DDH does not know $a$ (because it needs to embed the CDH/DDH challenge $g^a$ in the experiment), so $g^{ar}$ (for adversarially chosen $r$ such that $g^r$ is known to the reduction) looks random and the reduction cannot tell which $H$ query is $H(g^{ar})$ — which is overlooked in the proofs. This issue is particularly subtle since it emerges *after* $P'$ *outputs*, so one who thinks the reduction is done once $P'$ outputs would fail to detect it (this is where [BCP+23] fails). In addition, the issue is non-existent if the adversary modifies the first message but passes the second — in other words, the two messages are not "symmetric". Thus, a tight reduction to CDH in the latter case does not imply a tight reduction in the former case (this is where [DHP+18] fails).

Since most hybrid proofs for UC PAKE are complicated and involve a large number of hybrids, it might be helpful to provide a table similar to Tables 6 and 9 at the beginning — in addition to, or in lieu of, a prose summary of the chain of hybrids. We believe this helps the reader understand the essence of the proof without digging into the details of too many hybrids. Furthermore, (assuming the hybrids begin in the real world) it would be useful to include an argument on why the last hybrid is identical to the ideal world, with the challenger split into the simulator and the functionality; this is far from obvious in many cases.

**Sampling from $\mathbb{Z}_p$ and $\mathbb{Z}_p^*$ might make a difference.** Last but not least, we wish to bring up a point which shows that even seemingly minor issues might become significant in some contexts and thus should not be ignored. Let us start from a topic that appears unrelated. Suppose we have a group of prime order $p$. Recall that the Square Diffie-Hellman (SDH) assumption says that given $g^a$ where $a \leftarrow \mathbb{Z}_p$, it is hard to compute $g^{a^2}$. A standard reduction to CDH (see, e.g., the proof of [FKL18, Theorem 3.1]) works as follows: the reduction, on $A = g^a$, samples $r \leftarrow \mathbb{Z}_p$ and feeds $(A, A^r)$ to the CDH solver; upon receiving $X$ from the CDH solver, the reduction outputs $X^{\frac{1}{r}}$.

However, this reduction fails in the case of $r = 0$, which happens with probability $1/p$. A complete description of the reduction should say that it aborts if $r = 0$ and thus loses an additive term $1/p$, which is generally missing in existing works. Alternatively, the reduction could sample $r \leftarrow \mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$ — which is a reduction between variants of CDH and SDH where $a, b$ are also sampled from $\mathbb{Z}_p^*$.[27]

A similar issue appears in the 2HashDH Oblivious PRF (OPRF) [JKK14, JKKX16, JKKX17, JKX18, HJKW23, DFG+23], which (in its simplest form) involves two parties jointly evaluating the function $f_k(x) = H_2(x, H_1(x)^k)$ on some input $x$: a user, that holds $x$, samples $r \leftarrow \mathbb{Z}_p$ and sends $A := H_1(x)^r$ to a server; the server, that holds $k$, sends $B := A^k$ to the user; finally, the user outputs

[27] A better reduction would sample $r \leftarrow \mathbb{Z}_p$ and feed $(A, Ag^r)$ to the CDH solver, and upon receiving $X$ output $X/A^r$. This reduction works even if $r = 0$. Another advantage is that this reduction works nicely even in a composite-order cyclic group.

$H_2(x, B^{\frac{1}{r}})$. Obviously the user fails if $r = 0$, and it should instead sample $r \leftarrow \mathbb{Z}_p^*$. All of the cited works on the 2HashDH OPRF have the user sample $r \leftarrow \mathbb{Z}_p$, and since this protocol is used as a building block in various protocols (including the OPAQUE strong asymmetric PAKE protocol that has been recommended for standardization by IETF [JKX18]), this issue has dragged on for years.

While it might appear pedantic to insist on the edge case of $r = 0$, it turns out that sampling from $\mathbb{Z}_p$ and $\mathbb{Z}_p^*$ might actually make an essential difference, as we have seen in the attack on OEKE in Sect. 3.4. Recall again that the attack works as follows: the adversary sends $B = e$ and $\tau = F_e(1)$ to $P$, causing $P$ to output $F_e(0)$ — which the adversary can predict. This issue is somewhat hidden as sampling from $\mathbb{Z}_p^*$ is indeed unnecessary for the standard security notion of Diffie-Hellman; however, the key point is that in OEKE *we are using Diffie-Hellman in a somewhat non-standard manner, namely in the context where there is a man-in-the-middle adversary in the higher-level protocol that can in particular control the message* $g^b$. This means that missing the seemingly $1/p$ probability of sampling $b = 0$ actually "translates to" missing the collision resistance property of the KA protocol, which is necessary for the security of OEKE.

In general, it seems "safer" to always sample from $\mathbb{Z}_p^*$, as this excludes the $r = 0$ case that might cause us trouble. However, we believe it is warranted to develop a thorough understanding of whether sampling from $\mathbb{Z}_p$ and sampling from $\mathbb{Z}_p^*$ — or other seemingly minor issues — make an essential difference in a certain context, and if so, what the reason exactly is (see, e.g., [MX23, Footnote 12]). This might help reveal some general patterns that are hidden otherwise, such as collision resistance in our case (cf. Footnote 26).

**Remark 6.2.** *The case of the 2HashDH OPRF in [HJKW23, DFG+23] has a slightly more significant (but still minor) issue: they only require the GDH or the one-more GDH assumption in a cyclic group, without any specific requirements on the order. (Earlier works require the order to be prime.) This means that there might be more than one $r$-th root of $B = H_1(x)^{kr}$, and the user might use a wrong one (i.e., other than $H_1(x)^k$) in the OPRF output — even assuming that the $r$-th root(s) exist and can be efficiently computed. As a concrete example, say the group order is $2p$ where $p$ is prime (e.g., $\mathbb{Z}_q^*$ where $q$ is a strong prime). Assume the user's exponent $r \leftarrow \mathbb{Z}_{2p}$ is even, which happens with probability 1/2. If $r = 0$ then the $r$-th root of $B$ does not exist, so the user fails. Otherwise there are two $r$-th roots of $B$, $H_1(x)^k$ and $H_1(x)^{k+p}$, and the user has a 1/2 chance of using the wrong one in the OPRF output (note that if $k \leftarrow \mathbb{Z}_{2p}$ then which one is the right value is independent of the user's view). Overall this OPRF protocol has an over 1/4 correctness failure probability.*[28] *However, this is unrelated to our main point.*

## 6.4 Future Work

The first future direction that comes to mind is that in recent years there have been a number of *asymmetric* PAKE (aPAKE) protocols using IC or HIC [GJK21, SGJK22, SGJ23], and it would be interesting to see whether the (H)IC there can be replaced by the more efficient POPF as well. We expect the security analyses to be significantly more complicated than the one in this work, as those aPAKE protocols are built from Authenticated Key Exchange (AKE) rather than KA, and defining the security of AKE (let alone any additional properties necessary in the context of aPAKE) is a much more complicated and delicate task.

Another direction is to explore the quantum security of (O)EKE. While Kyber is a post-quantum KA protocol, the POPF in (O)EKE uses an RO, to which our analysis only considers classical queries. As such, our analysis serves as a first step towards the post-quantum security of (O)EKE,

[28] Interestingly, another section of [DFG+23] correctly requires the group order to be prime (see Theorem 3 in Appendix D). But their analysis of the 2HashDH OPRF (Theorem 2 in Appendix C.2) does not make this requirement.

but a complete analysis would at least require working in the Quantum-accessible ROM (QROM) [BDF+11], rather than the ROM. One difficulty is that the study of QROM and the study of UC have been developed mostly in parallel; while there have been works proving the universal composability theorem with a quantum environment [Unr10], to the best of our knowledge there have been no works on formalizing the QROM in the quantum UC framework. It seems that some theoretical foundations need to be laid out before we can pursue this direction.

## References


[ABB+20] Michel Abdalla, Manuel Barbosa, Tatiana Bradley, Stanislaw Jarecki, Jonathan Katz, and Jiayu Xu. Universally composable relaxed password authenticated key exchange. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 278–307. Springer, Heidelberg, August 2020.

[AHH21] Michel Abdalla, Björn Haase, and Julia Hesse. Security analysis of CPace. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 711–741. Springer, Heidelberg, December 2021.

[AP05] Michel Abdalla and David Pointcheval. Simple password-based encrypted key exchange protocols. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 191–208. Springer, Heidelberg, February 2005.

[AWZ23] Damiano Abram, Brent Waters, and Mark Zhandry. Security-preserving distributed samplers: How to generate any CRS in one round without random oracles. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 489–514. Springer, Heidelberg, August 2023.

[BCP03] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Security proofs for an efficient password-based key exchange. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 241–250. ACM Press, October 2003.

[BCP+23] Hugo Beguinet, Céline Chevalier, David Pointcheval, Thomas Ricosset, and Mélissa Rossi. GeT a CAKE: Generic transformations from key encaspulation mechanisms to password authenticated key exchanges. In Mehdi Tibouchi and Xiaofeng Wang, editors, *ACNS 23, Part II*, volume 13906 of *LNCS*, pages 516–538. Springer, Heidelberg, June 2023.

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, Heidelberg, December 2011.

[BFGJ17] Jacqueline Brendel, Marc Fischlin, Felix Günther, and Christian Janson. PRF-ODH: Relations, instantiations, and impossibility results. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 651–681. Springer, Heidelberg, August 2017.

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Heidelberg, May 2000.

[CHK+05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, Heidelberg, May 2005.

[Cry20] Crypto Forum Research Group. PAKE selection, 2020. https://github.com/cfrg/pake-selection.

[DFG+23] Gareth T. Davies, Sebastian H. Faller, Kai Gellert, Tobias Handirk, Julia Hesse, Máté Horváth, and Tibor Jager. Security analysis of the WhatsApp end-to-end encrypted backup protocol. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 330–361. Springer, Heidelberg, August 2023.

[DHP+18] Pierre-Alain Dupont, Julia Hesse, David Pointcheval, Leonid Reyzin, and Sophia Yakoubov. Fuzzy password-authenticated key exchange. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 393–424. Springer, Heidelberg, April / May 2018.

[DS16] Yuanxi Dai and John P. Steinberger. Indifferentiability of 8-round Feistel networks. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 95–120. Springer, Heidelberg, August 2016.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 537–554. Springer, Heidelberg, August 1999.

[GJK21] Yanqi Gu, Stanislaw Jarecki, and Hugo Krawczyk. KHAPE: Asymmetric PAKE from key-hiding key exchange. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 701–730, Virtual Event, August 2021. Springer, Heidelberg.

[GK10] Adam Groce and Jonathan Katz. A new framework for efficient password-based authenticated key exchange. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 516–525. ACM Press, October 2010.

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, Heidelberg, May 2003. https://eprint.iacr.org/2003/032.ps.gz.

[HJKW23] Julia Hesse, Stanislaw Jarecki, Hugo Krawczyk, and Christopher Wood. Password-authenticated TLS via OPAQUE and post-handshake authentication. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 98–127. Springer, Heidelberg, April 2023.

[HL19] Björn Haase and Benoît Labrique. Aucpace: Efficient verifier-based pake protocol tailored for the iiot. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 1–48, 2019.

[Jar23] Stanislaw Jarecki. Randomized half-ideal cipher on groups with applications to UC (a)PAKE. https://www.youtube.com/watch?v=GL4m7StDsPg, 2023. Talk at EUROCRYPT 2023.

[JKK14] Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 233–253. Springer, Heidelberg, December 2014.

[JKKX16] Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online). In *IEEE EuroS&P 2016*, pages 276–291. IEEE, March 2016.

[JKKX17] Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. TOPPSS: Cost-minimal password-protected secret sharing based on threshold OPRF. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17*, volume 10355 of *LNCS*, pages 39–58. Springer, Heidelberg, July 2017.

[JKSS12] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 273–293. Springer, Heidelberg, August 2012.

[JKX18] Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 456–486. Springer, Heidelberg, April / May 2018.

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 475–494. Springer, Heidelberg, May 2001.

[LLHG23] Xiangyu Liu, Shengli Liu, Shuai Han, and Dawu Gu. EKE meets tight security in the Universally Composable framework. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 685–713. Springer, Heidelberg, May 2023.

[MRR20] Ian McQuoid, Mike Rosulek, and Lawrence Roy. Minimal symmetric PAKE and 1-out-of-N OT from programmable-once public functions. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 425–442. ACM Press, November 2020.

[MX23] Ian McQuoid and Jiayu Xu. An efficient strong asymmetric pake compiler instantiable from group actions. In *ASIACRYPT 2023*, pages 176–207. Springer, December 2023.

[RX23] Lawrence Roy and Jiayu Xu. A universally composable PAKE with zero communication cost - (and why it shouldn't be considered UC-secure). In Alexandra Boldyreva and

Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 714–743. Springer, Heidelberg, May 2023.

[SAB+22] Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[SGJ23] Bruno Freitas Dos Santos, Yanqi Gu, and Stanislaw Jarecki. Randomized half-ideal cipher on groups with applications to UC (a)PAKE. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 128–156. Springer, Heidelberg, April 2023.

[SGJK22] Bruno Freitas Dos Santos, Yanqi Gu, Stanislaw Jarecki, and Hugo Krawczyk. Asymmetric PAKE with low computation and communication. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part II*, volume 13276 of *LNCS*, pages 127–156. Springer, Heidelberg, May / June 2022.

[Unr10] Dominique Unruh. Universally composable quantum multi-party computation. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 486–505. Springer, Heidelberg, May / June 2010.

# A EKE Is Insecure If the Underlying KA Is Not Strongly Pseudorandom

In this section we show that KA security and pseudorandomness combined do not imply strong pseudorandomness. We first present a general discussion showing the subtleties in the analysis, and then give a concrete counterexample. Finally we use a simple attack to demonstrate why strong pseudorandomness is necessary for the security of EKE. Please see Defs. 2.2 to 2.5 for the definitions of these properties.

**Indistinguishability between six distributions (but not the seventh).** In a KA protocol, each of the first message $A$, the second message $B$, and the key $K$ may be "real" or "random", resulting in 8 potential joint distributions of $(A, B, K)$. (See Table 12 for definitions of "real" and "random".) The case where $A, B$ are random but $K$ is real (henceforth (random $A$, random $B$, real $K$), or simply (random, random, real)) is not well-defined, since for $K$ to be real, either $a$ or $b$ needs to be defined (which implies that either $A$ or $B$ needs to be real). Are the remaining 7 distributions indistinguishable from each other?

| | real | random |
|---|---|---|
| $A$ | $a \leftarrow \mathcal{R}$<br>$A := \mathsf{msg}_1(a)$ | $A \leftarrow \mathcal{M}_1$ |
| $B$ | $b \leftarrow \mathcal{R}$<br>$B := \mathsf{msg}_2(b, A)$ | $B \leftarrow \mathcal{M}_2$ |
| $K$ | $K := \mathsf{key}_1(a, B)$<br>or<br>$K := \mathsf{key}_2(b, A)$ | $K \leftarrow \mathcal{K}$ |

Table 12: Definitions of "real" and "random" for $A, B, K$

Recall that first pseudorandomness says that real $A$ is indistinguishable from random $A$, and second pseudorandomness says that (real $A$, real $B$) is indistinguishable from (real $A$, random $B$). Note that the definition of real $B$ does not use $a$ (it only uses $A$) and thus can be simulated by a reduction to first pseudorandomness; this means that pseudorandomness (first and second combined) implies that the 4 joint distributions of $A$ and $B$ — where both $A$ and $B$ might be real or random — are indistinguishable from each other. Since random $K$ can be simulated without any knowledge about $A$ or $B$, pseudorandomness implies that the following 4 distributions are indistinguishable from each other:

- (real, real, random),
- (real, random, random),
- (random, real, random),
- (random, random, random).

Security says that (real, real, random) is indistinguishable from (real, real, real), so now we have 5 indistinguishable distributions under pseudorandomness plus security. Furthermore, (random, real, real) is indistinguishable from (real, real, real); this is because a reduction to first pseudorandomness can sample $b$ on its own and simulate both real $B$ and real $K$ without knowing $a$. (This immediately

implies Lem. 5.2, which says that (random, real, real) is indistinguishable from (random, random, random).) In summary, 6 out of the 7 distributions are indistinguishable from each other.

What about the last distribution, (real, random, real)? Suppose we attempt to construct a reduction that shows the indistinguishability from (real, real, real). The reduction, on (real $A$, real $B$) or (real $A$, random $B$), needs to simulate real $K$ — which it cannot do because it knows *neither* $a$ (which is used in real $A$) *nor* $b$ (which is used in real $B$). What we need here is that (real $A$, real $B$) and (real $A$, random $B$) are indistinguishable *even given real* $K$, which is exactly *strong* second pseudorandomness.

In summary, security plus strong pseudorandomness imply the indistinguishability between all 7 distributions of $(A, B, K)$ (which in particular implies Lem. 5.5 which says that (real, random, real) is indistinguishable from (real, random, random)); whereas security plus pseudorandomness only imply the indistinguishability between 6 distributions of $(A, B, K)$, with (real, random, real) excluded.

**Remark A.1.** *If the KA protocol is 1-simultaneous round, i.e.,* $B = \mathsf{msg}_2(b)$ *does not depend on* $A$*, then strong pseudorandomness is implied by pseudorandomness: the reduction, on real* $B$ *or random* $B$*, can sample* $a$ *on its own and simulate both real* $A$ *and real* $K$ *without knowing* $b$*. [SGJ23, Theorem 2] only considers 1-simultaneous round KAs, so the distinction between pseudorandomness and strong pseudorandomness does not exist there; whereas [MRR20, Theorem 10] considers general 2-round KAs and overlooks this subtlety.*

**Counterexample.** We now present a concrete counterexample to show that security and pseudorandomness combined indeed do not imply the indistinguishability between (real, random, real) and the other 6 distributions. Consider a variant of hashed Diffie-Hellman, where

$$\begin{aligned}\mathsf{msg}_1(a) &= g^a\\ \mathsf{msg}_2(b, A) &= (g^b, H_0(A^b))\\ \mathsf{key}_2(b, A) &= H_1(A^b)\end{aligned}$$

(where $H_0, H_1 \colon \mathbb{G} \to \{0,1\}^\kappa$ are ROs). In this variant, $P$ (that holds $a$) can use the $H_0$ hash to check whether it received a valid response to $A$; we use this to break strong pseudorandomness. Let

$$\mathsf{key}_1(a, (B_0, B_1)) = \begin{cases} H_1(B_0^a) & \text{if } B_1 = H_0(B_0^a)\\ H_2(g^a) & \text{otherwise}\end{cases}$$

Correctness is easily verified. Security and pseudorandomness still hold, as $g^a$ and $g^b$ are uniform elements of $\mathbb{G}$, and $H_0(A^b)$ and $H_1(A^b)$ are indistinguishable from random strings assuming CDH. However, (real, random, real) and (random, random, random) are easily distinguishable: in the former distribution, $K = H_2(A)$ because $B_1$ is uniform and so has negligible chance of satisfying $B_1 = H_0(B_0^a)$, while in the latter $K \neq H_2(A)$ with overwhelming probability, as they are independently random $\kappa$-bit strings.

**Necessity of strong pseudorandomness in EKE.** Consider the following simple attack on EKE (using IC): the adversary disregards $P'$, and on $\phi$ from $P$ sends random $\phi^*$ to $P$. After $P$ outputs $K$, the adversary queries $A := \mathcal{D}(\mathsf{pw}, \phi)$ and $B := \mathcal{D}(\mathsf{pw}, \phi^*)$ (where $\mathsf{pw}$ is the password of $P$).

In the real world, $A$ is the KA message generated by the honest $P$, so $A$ is real; $B$ is the IC decryption of random $\phi^*$, so $B$ is random; and $K$ is computed by the honest $P$ on $a$ and $B$, so

$K$ is real. In other words, the environment's view is (real, random, real). Now consider the ideal world: before $P$ outputs $K$ the simulator only sees a random $\phi^*$ from the adversary and has no knowledge about $\mathsf{pw}$, so it cannot send a correct $\mathsf{TestPwd}$ command and thus $\mathcal{F}_{\text{PAKE}}$ will set the session key of $P$ to be random. (The simulator sees $\mathsf{pw}$ *after* $P$ *outputs*, at which time it cannot do anything to change the session key of $P$.) That is, in the ideal world the environment's view is ($\star$, $\star$, random), where $\star$ could be real or random, depending on the simulator's strategy. However, we have just seen that without strong pseudorandomness (real, random, real) is distinguishable from *all other 6 distributions*, so no matter what the two $\star$ are, the ideal world and the real world are distinguishable (the environment simply runs the distinguisher between (real, random, real) and the appropriate ($\star$, $\star$, random) for KA).

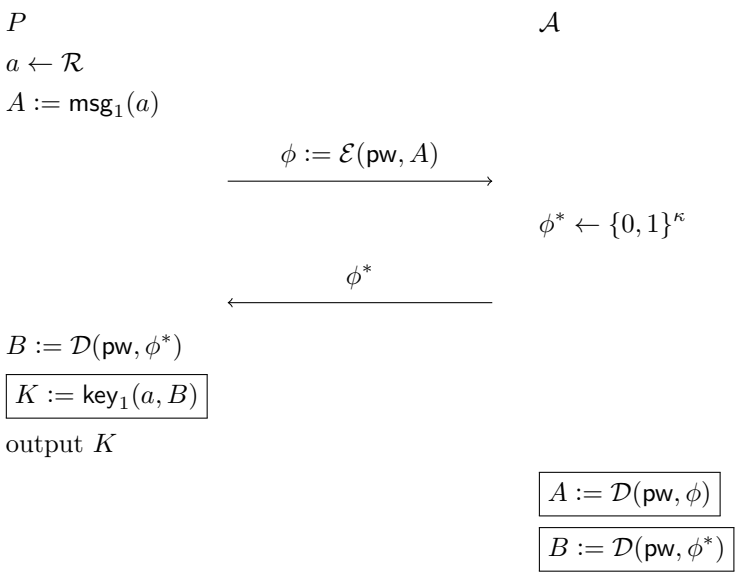


Figure 23: Attack on EKE with a KA protocol that does not satisfy strong pseudorandomness. In the real world $\mathcal{Z}$ sees (real $A$, random $B$, real $K$) (boxed texts). In the ideal world the simulator knows $\mathsf{pw}$ only after $P$ outputs, so $K$ is random. Without strong pseudorandomness, (real $A$, random $B$, real $K$) is distinguishable from ($\star$, $\star$, random $K$)

The above shows that [MRR20, Theorem 10] is false in yet another way, since it claims the security of EKE without requiring strong pseudorandomness for KA. Furthermore, the issue appears to persist even if $K$ is hashed, as security plus pseudorandomness do not even seem to imply the unpredictability of real $K$ given real $A$ and random $B$. This suggests that there might also be an additional flaw in [BCP+23, Theorem 1](we are not able to identify the exact flaw in the proof). Also note that the attack above does not apply to OEKE, where the adversary needs to come up with a valid authenticator in order for $P$ to output a real session key.

# B EKE and OEKE Are Insecure If the Underlying KA Is Not Pseudorandom Non-Malleable

In this section we present a counterexample showing that KA security, strong pseudorandomness, and non-malleability combined do not imply pseudorandom non-malleability. By non-malleability, we mean that it satisfies a variant of Def. 3.6 where $B$ is sampled as a real KA message in both

experiments, i.e., the following two distributions are indistinguishable:

| | |
|---|---|
| $a \leftarrow \mathcal{R}$ | $a \leftarrow \mathcal{R}$ |
| $b \leftarrow \mathcal{R}$ | $b \leftarrow \mathcal{R}$ |
| $A := \mathsf{msg}_1(a)$ | $A := \mathsf{msg}_1(a)$ |
| $B := \mathsf{msg}_2(b, A)$ | $B := \mathsf{msg}_2(b, A)$ |
| $K' := \mathsf{key}_2(b, A)$ | $K' \leftarrow \mathcal{K}$ |
| $B^* \leftarrow \mathcal{A}(A, B, K')$ | $B^* \leftarrow \mathcal{A}(A, B, K')$ |
| abort if $B^* = B$ | abort if $B^* = B$ |
| $K := \mathsf{key}_1(a, B^*)$ | $K := \mathsf{key}_1(a, B^*)$ |
| output $K$ to $\mathcal{A}$ | output $K$ to $\mathcal{A}$ |

We then show that both EKE and OEKE with our KA protocol are insecure, demonstrating why pseudorandom non-malleability is necessary for the security of (O)EKE.[29]

**Counterexample.** The counterexample is similar to the one in Appx. A, using an additional field of the message to detect improperly generated or modified messages.

$$\begin{aligned}
\mathsf{msg}_1(a) &= g^a \\
\mathsf{msg}_2(b, A) &= (g^b, H_0(A^b)) \\
\mathsf{key}_1(a, (B_0, B_1)) &= \begin{cases} H_1(B_0^a) & \text{if } B_1 = H_0(B_0^a) \\ H_2(g^a) & \text{if } B_1 = H_0(B_0^a) - 1 \\ H_3(a, B) & \text{otherwise} \end{cases} \\
\mathsf{key}_2(b, A) &= H_1(A^b)
\end{aligned}$$

(where $H_0, H_1, H_2 \colon \mathbb{G} \to \{0,1\}^\kappa, H_3 \colon \mathbb{Z}_p \times \mathbb{G} \to \{0,1\}^\kappa$ are ROs).

Correctness, security, and pseudorandomness all hold assuming CDH, for the same reasons as with the previous counterexample. Additionally, strong pseudorandomness holds because a uniformly random $B$ triggers the $H_3(a, B)$ case with overwhelming probability, which is indistinguishable from random because $a$ cannot be guessed by the adversary (without solving discrete log); and a real $B$ triggers the $H_1(B_0^a)$ case, which is also indistinguishable from random by CDH.[30]

Now let's see why pseudorandom non-malleability fails. The adversary $\mathcal{A}$ receives $(A, B, K')$ (where $B = (B_0, B_1)$) from the challenger, and outputs $B^* = (B_0, B_1 + 1)$. The challenger then sends $K = \mathsf{key}_1(a, B^*)$. In the real distribution this will trigger the second case and so $K = H_2(A)$, while in the random distribution it will trigger the third case and output $H_3(a, B)$ — because in the random distribution $B_1$ is uniform, so $B_1 + 1$ is as well, so both of them have negligible chance of equaling $H_0(B_0^a)$. To distinguish, $\mathcal{A}$ simply checks whether $K = H_2(A)$.

**Attacks on EKE and OEKE.** This counterexample also works on the whole EKE protocol, not just the pseudorandom non-malleability definition. Below we illustrate the attack, where the adversary behaves similarly to the pseudorandom non-malleability experiment: it passes the $P$-to-$P'$

[29] Recall that Sects. 3.1 to 3.3 already show that non-malleability is necessary for the security of EKE and OEKE-PRF. Here we additionally show that (1) the stronger *pseudorandom* non-malleability is needed, and (2) *any* version of OEKE, including OEKE-RO, needs pseudorandom non-malleability.

[30] The KA protocol can be additionally made collision-resistant by changing the ROs $H_1, H_2, H_3$ to have $3\kappa$-bit outputs and requiring $B_0$ to not be the identity element, since all three cases of $\mathsf{key}_1$ hash something that uniquely identifies $A$.

message without modification, and on the $P'$-to-$P$ message $\phi'$, the adversary decrypts using $\mathsf{pw}$ (the password of $P$) and obtains $(B_0, B_1)$, then encrypts $(B_0, B_1 + 1)$. If $\mathsf{pw}'$ (the password of $P'$) is equal to $\mathsf{pw}$, then $(B_0, B_1) = (g^b, H_0(g^{ab}))$, so $\mathsf{key}_1$ will enter the second case and $P$ will output the predictable value $H_2(g^a)$; otherwise $(B_0, B_1)$ is uniformly random, so with overwhelming probability $\mathsf{key}_1$ will enter the third case and $P$ will output $H_3(a, (B_0, B_1 + 1))$. In other words, the enviroment/adversary can check if the passwords of $P$ and $P'$ match by modifying the $P'$-to-$P$ message and observing the session key of $P$ only — which is not allowed by the UC-security of PAKE (throughout the entire experiment the simulator never knows $\mathsf{pw}'$, so it cannot make any decision based on whether $\mathsf{pw} = \mathsf{pw}'$ or not). Note that this attack requires the adversary to know $\mathsf{pw}$ but not $\mathsf{pw}'$; and unlike other attacks where the adversary passes the first message and changes the second (Sects. 3.1, 3.2 and 3.5), here the environment does not need to observe the session key of $P'$.

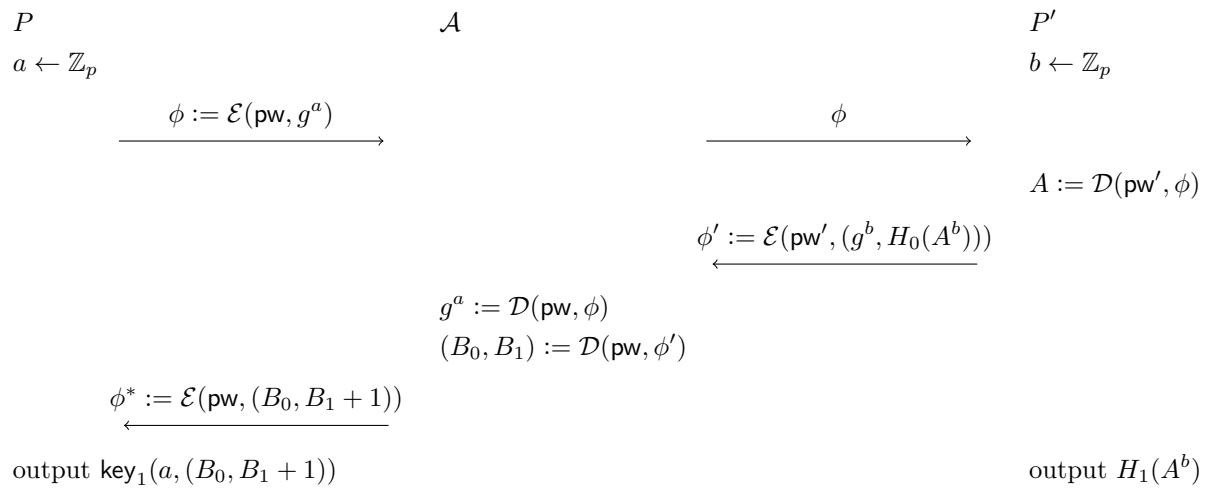


Figure 24: Attack on EKE with the KA protocol in this section (which is not pseudorandom non-malleable). $\mathcal{A}$ causes $P$ to output the predictable $H_2(g^a)$ if $\mathsf{pw} = \mathsf{pw}'$ and the unpredictable $H_3(a, (B_0, B_1 + 1))$ otherwise, without attacking the $P'$ session. This is not allowed by UC-security

A similar attack works against OEKE, with the adversary again modifying only the second ($P'$-to-$P$) message. For OEKE, the adversary receives $(B_0, B_1, \tau')$ from $P'$, and changes them to $(B_0, B_1 + 1, \tau^*)$, where $K^* \| \tau^* := H_2(g^a)$ (this requires the adversary to know $\mathsf{pw}$ and decrypt the $P$-to-$P'$ message). If $\mathsf{pw} = \mathsf{pw}'$, we have $\mathsf{key}_1(a, (B_0, B_1 + 1)) = H_2(g^a)$, so $\tau^* = \tau$ and $P$ will output $K^*$. If $\mathsf{pw} \neq \mathsf{pw}'$, with overwhelming probability $\mathsf{key}_1(a, (B_0, B_1 + 1)) = H_3(a, (B_0, B_1))$ which has negligible probability of matching $\tau^*$, so the session key of $P$ is uniformly random with overwhelming probability. Again, the enviroment/adversary can check if the passwords of $P$ and $P'$ match by modifying the $P'$-to-$P$ message and observing the session key of $P$ only.

**Non-malleability.** Given how badly EKE and OEKE break when using this KA protocol, it might seem that non-malleability is the problem. However, our KA protocol does satisfy non-malleability under CDH, as we will prove now. It is just pseudorandom non-malleability that is the problem.

The reduction to CDH, on challenge $A = g^a$ and $B = g^b$, samples $K' \leftarrow \{0,1\}^\kappa$ and sends $(A, B, K')$ to the non-malleability adversary $\mathcal{A}$. Next, $\mathcal{A}$ outputs some $B^* \neq B$, and the reduction responds with some $K$ according to the following rules:

- If $B_0^* = B_0$ and $B_1^* = B_1 + 1$, respond with $H_2(A)$.

- If $B_1^* = H_0(X)$ for some past $H_0(X)$ query, sample a random bit $d \leftarrow \{0,1\}$ to decide whether to treat it as a correct guess. If $d = 1$, output $H_1(X)$.
- If $B_1^* = H_0(X) + 1$ for some past $H_0(X)$ query, sample a random bit $d \leftarrow \{0,1\}$ to decide whether to treat it as a correct guess. If $d = 1$, output $H_2(A)$.
- If none of these cases matches, or if $d = 0$, output $K \leftarrow \{0,1\}^\kappa$.

So far, we have a nearly perfect simulation of the random $K'$ distribution. The only exceptions are that $\mathcal{A}$ could guess $a$ in a $H_3$-query, but we could then use $a$ to solve CDH; and the (negligible) chance that there is a collision in $H_0$ or that $\mathcal{A}$ will find a new $H_0$ pre-image to $B_1^*$ after it was already sent to the reduction.

In order for $\mathcal{A}$ to distinguish this from the real $K'$ distribution, it must make a query to the preimage $H_1(g^{ab})$. Therefore, the reduction guesses a random $H_1(X)$ query, and outputs $X$ in the CDH experiment. If $\mathcal{A}$ has advantage $\mathsf{Adv}_{\mathrm{NM}}$ and makes $q$ queries to $H_1$, then the reduction has advantage

$$\mathsf{Adv}_{\mathrm{CDH}} \geq \frac{1}{2q}\mathsf{Adv}_{\mathrm{NM}} - \mathrm{negl.},$$

since the reduction wins when both guesses ($d$ and the $H_1$ query) are correct, which have probabilities $1/2$ and $1/q$, respectively.

The above shows a crucial difference between non-malleability and pseudorandom non-malleability: non-malleability can be obtained by taking any secure and pseudorandom KA protocol and applying an RO hash to the key, although the reduction is loose (an extension of non-malleability of hashed Diffie-Hellman under DDH); whereas pseudorandom non-malleability is not implied by any existing properties and must be presented as a property on its own. This in particular means that [BCP+23, Theorem 1] and [BCP+23, Theorem 2] are false: while the (O)EKE protocols there hashes the KA key at the end, this only guarantees non-malleability, and the theorems do not mention pseudorandom non-malleability which is actually needed.

# C Properties of Kyber

We now argue that Kyber (Sect. 2.1) satisfies correctness, security, strong pseudorandomness, pseudorandom non-malleability, and collision resistance — properties that are needed for the security of (O)EKE.

For correctness, the idea is that $\mathsf{Kyber.Dec}(a, B) = b$, so $\mathsf{key}_1$ will see $B$ as being a valid message (i.e., $B = \mathsf{msg}_2(b, A)$) and will output $H'(b, A, B)$, the same as $\mathsf{key}_2$. Security and pseudorandomness follow easily from the corresponding properties of the underlying public-key encryption scheme: CPA-security, pseudorandom public keys, and pseudorandom ciphertexts.

By applying the FO transform we have added the properties of strong pseudorandomness, pseudorandom non-malleability, and collision resistance. Collision resistance is the easiest to see: $\mathsf{key}_1$ always outputs either $H'(b, A, B)$ or $H'(a, B)$, and in both cases something that uniquely identifies $A$ is included in the RO. Therefore, collision resistance holds as long as $H'$ has sufficient output length for $\tau$ to be at least $2\kappa$-bit long.

For strong pseudorandomness, notice that a uniformly random ciphertext $B$ has negligible probability of equaling $\mathsf{msg}_2(\mathsf{Kyber.Dec}(a, B), A)$, since $\mathsf{msg}_2$ is defined using $H(b, A)$, and this will be a fresh RO query. (Technically, this requires $\mathsf{Kyber.Enc}$ to preserve sufficient entropy from $H(b, A)$; this is true for Kyber.) Therefore, $\mathsf{key}_1(a, B) = H'(a, B)$ on uniformly random $B$, so the key is indistinguishable from uniformly random. This shows that (real $A$, random $B$, real $K$) is

indistinguishable from (real $A$, random $B$, random $K$), which (by Appx. A) is sufficient to show strong pseudorandomness.

Finally, we need to show pseudorandom non-malleability. Normal non-malleability is implied by the CCA-security of the FO transform, which suggests that we should look for a similar argument to show pseudorandom non-malleability. The main idea of the FO transform is that on any adversarially generated $B$, one can look through all $H(b, A)$ queries made by the adversary and see whether it matches any of them. If it does, then we already know $b$, and can simulate the output of $\mathsf{key}_1$ without knowing $a$ by just computing $H'(b, A, B)$. If there is no such query, then there is negligible probability that $B = \mathsf{msg}_2(\mathsf{Kyber.Dec}(a, B), A)$, as either $H(\mathsf{Kyber.Dec}(a, B), A)$ is a fresh query and will add enough entropy to $\mathsf{msg}_2$ to stop it from matching $B$, or it is one of the previous $H(b, A)$ queries that are already known to not produce the correct $B$. In that case, $\mathsf{key}_1$ can be simulated by just outputting a uniformly random value, since $H'(a, B)$ is indistinguishable from random. Therefore, the FO transform allows any $\mathsf{key}_1$ to be simulated without knowledge of $a$, and without $\mathsf{key}_1$ pseudorandom non-malleability is just a combination of security and pseudorandomness.

# D Additional Results on the Security of EKE

In this section we give proof sketches for the security of ("raw") EKE. Recall that in Thm. 5.4 we only proved the UC-security of EKE-PRF (where the session key is $F_K(\phi')$ rather than the KA key $K$), and that is because EKE only realizes a weaker functionality, $\mathcal{F}_{\text{PAKE-sp}}$, if instantiated with POPF — as demonstrated by the attack in Sect. 3.5. Below we argue for two results:

1. EKE using IC realizes $\mathcal{F}_{\text{PAKE}}$; and
2. EKE using POPF realizes $\mathcal{F}_{\text{PAKE-sp}}$,

which correspond to items (1) and (2) in Sect. 3.5. In both results, the vast majority of the proof is identical to that of Thm. 5.4, so we only highlight the differences. We begin with (2) since the proof of (1) is almost immediate given the proof of (2).

## D.1 EKE Using POPF Realizes $\mathcal{F}_{\text{PAKE-sp}}$

We first present the $\mathcal{F}_{\text{PAKE-sp}}$ functionality in Figure 25. The only difference with $\mathcal{F}_{\text{PAKE}}$ is that the ideal adversary can additionally send a TestSamePwd command, which works exactly as TestPwd except that if $\mathsf{pw}^* = \mathsf{pw} = \mathsf{pw}'$ (i.e., the password guess is correct and the two parties' passwords match) then the functionality does not do anything.

Consider the EKE protocol, which is the EKE-PRF protocol in Figure 19 except that $P$ outputs $K$ instead of $\mathsf{PRF}_K(\phi')$, and $P'$ outputs $K'$ instead of $\mathsf{PRF}_{K'}(\phi')$. We have:

**Theorem D.1.** *The EKE protocol realizes $\mathcal{F}_{\text{PAKE-sp}}$ in the $(\mathcal{F}_{\text{POPF}}, \mathcal{F}_{\text{PAKE-1r}})$-hybrid world.*

*Proof (sketch).* The simulator $\mathcal{S}$ is shown in Figure 26.

The only modifications from the EKE-PRF simulator in Figure 20 are:

- In a NewKey command, we replace $\mathsf{PRF}_{K^*}(\phi')$ or $\mathsf{PRF}_{K^*}((\phi')^*)$ with $K^*$;
- We add case 12(3)(i), where a TestSamePwd command instead of TestPwd is sent.

The first modification is consistent with what changes in the real world, namely parties output $K$ (resp. $K'$) rather than $\mathsf{PRF}_K((\phi')^*)$ (resp. $\mathsf{PRF}_{K'}(\phi')$). For the second modification, recall that TestSamePwd is exactly the same as TestPwd, except that in the case of $(\mathsf{pw}')^* = \mathsf{pw} = \mathsf{pw}'$,

- On input $(\mathsf{NewSession}, \mathsf{sid}, P, P', \mathsf{pw}, \mathsf{role})$ from $P$, send $(\mathsf{NewSession}, \mathsf{sid}, P, P', \mathsf{role})$ to $\mathcal{S}$. Furthermore, if this is the first NewSession message for *sid*, or this is the second NewSession message for sid and there is a record $\langle P', P, \cdot \rangle$, then record $\langle P, P', \mathsf{pw} \rangle$ and mark it fresh.
- On $(\mathsf{TestPwd}, \mathsf{sid}, P, \mathsf{pw}^*)$ from $\mathcal{S}$, if there is a record $\langle P, P', \mathsf{pw} \rangle$ marked fresh, then do:
  - If $\mathsf{pw}^* = \mathsf{pw}$, then mark the record compromised and send "correct guess" to $\mathcal{S}$.
  - If $\mathsf{pw}^* \neq \mathsf{pw}$, then mark the record interrupted and send "wrong guess" to $\mathcal{S}$.
- On $(\mathsf{TestSamePwd}, \mathsf{sid}, P, \mathsf{pw}^*)$ from $\mathcal{S}$, if there is a record $\langle P, P', \mathsf{pw} \rangle$ marked fresh, then do:
  - If $\mathsf{pw}^* = \mathsf{pw} = \mathsf{pw}'$, then ignore the command (and the record remains fresh).
  - If $\mathsf{pw}^* = \mathsf{pw} \neq \mathsf{pw}'$, then mark the record compromised and send "correct guess" to $\mathcal{S}$.
  - If $\mathsf{pw}^* \neq \mathsf{pw}$, then mark the record interrupted and send "wrong guess" to $\mathcal{S}$.
- On $(\mathsf{NewKey}, \mathsf{sid}, P, K^* \in \{0,1\}^\kappa)$ from $\mathcal{S}$, if there is a record $\langle P, P', \mathsf{pw} \rangle$, and this is the first NewKey message for *sid* and $P$, then output $(\mathsf{sid}, K)$ to $P$, where $K$ is defined as follows:
  - If the record is compromised, or either $P$ or $P'$ is corrupted, then set $K := K^*$.
  - If the record is fresh, a key $(\mathsf{sid}, K')$ has been output to $P'$, at which time there was a record $\langle P', P, \mathsf{pw} \rangle$ marked fresh, then set $K := K'$.
  - Otherwise sample $K \leftarrow \{0,1\}^\kappa$.

  Finally, mark the record completed.

Figure 25: UC PAKE with same password test functionality $\mathcal{F}_{\text{PAKE-sp}}$

$\mathcal{F}_{\text{PAKE-sp}}$ on TestSamePwd does nothing. This means that the only essential difference between the ideal world here and the ideal world in the proof of Thm. 5.4, is that if

$$\mathsf{pw} = \mathsf{pw}' \wedge \mathsf{pw}^* = \bot \wedge (\phi')^* \neq \phi' \wedge B^* = B \wedge (\phi')^* \text{ contains the correct password guess,}$$

we now let $P$ and $P'$ output the same session key. (This is the re-encryption case in Sect. 3.5 and the proof of Thm. 5.4.) This again matches what changes in the real world, where in this case the session keys of $P$ and $P'$ are indeed the same.

Given the intuition above, we can easily come up with a series of hybrids that are very similar to those in the proof of Thm. 5.4, with only two modifications:

- Wherever $\mathsf{PRF}_K(\phi')$ (resp. $\mathsf{PRF}_{K'}(\phi')$ or $\mathsf{PRF}_{K^*}(\phi')$) is mentioned, it is replaced by $K$ (resp. $K'$ or $K^*$). Furthermore, the following hybrids that trivially reduce to the fact that PRF is a PRF are now removed:
  - Hybrid 5 which sets the two session keys equal when $K = K'$;
  - Hybrid 6 which replaces $\mathsf{PRF}_{K'}$ (where $K'$ is a random string independent of the rest of the experiment) with a random function $G'$; and
  - Hybrid 7 which replaces $\mathsf{PRF}_K$ (where $K$ is a random string independent of the rest of the experiment) with a random function $G$.
- Hybrid 8 in the previous proof — which exactly deals with the re-encryption case above, and the change is that $P$ and $P'$ output session keys $(\mathsf{PRF}_K((\phi')^*), G'(\phi'))$ rather than $(G'((\phi')^*), G'(\phi'))$ — is also removed. The reason is slightly different than the above, though;

Initialize $U := \{\}$ as the set of $H$-evaluations.
Let $T = \{\}$ be the record of honest POPF evaluations as in $\mathcal{F}_{\text{POPF}}$.

To compute $H(x)$:

1. If there is an entry $(x, y) \in U$ return $y$.
2. Otherwise sample $y \leftarrow \mathcal{R}$, add $(x, y)$ to $U$ and return $y$.

On $(\mathsf{NewSession}, \mathsf{sid}, P, P', \text{“initiator”})$ from $\mathcal{F}_{\text{PAKE}}$:

3. Send $(\mathsf{Program}, \mathsf{sid}, P, P')$ from $\mathcal{F}_{\text{PAKE-1r}}$ to $\mathcal{A}$.

On $(\mathsf{NewSession}, \mathsf{sid}, P', P, \text{“respondent”})$ from $\mathcal{F}_{\text{PAKE}}$:

4. Send $(\mathsf{SampleResp}, \mathsf{sid}, P', P)$ from $\mathcal{F}_{\text{PAKE-1r}}$ to $\mathcal{A}$.

On $(\mathsf{Eval}, \mathsf{sid}, P, P', x)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{PAKE-1r}}$:

5. Compute $A = \mathsf{msg}_1(H(x))$ and return $A$ to $\mathcal{A}$.

On $(\mathsf{Deliver}, \mathsf{sid}, P, P', \mathsf{pw}^*, A^*)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{PAKE-1r}}$:

6. Send $(\mathsf{Program}, \mathsf{sid})$ from $\mathcal{F}_{\text{POPF}}$ to $\mathcal{A}$ and wait until $\mathcal{A}$ responds with $(\mathsf{Program}, \mathsf{sid}, \phi')$ to $\mathcal{F}_{\text{POPF}}$ such that there is no entry $(\phi', \cdot, \cdot) \in T$.
7. Send $(\mathsf{sid}, \phi')$ from $P'$ to $P$.
8. If $\mathsf{pw}^* = \bot$, send $(\mathsf{NewKey}, \mathsf{sid}, P', 0^\kappa)$ to $\mathcal{F}_{\text{PAKE}}$.
9. If $\mathsf{pw}^* \neq \bot$ do:
   (1) Send $(\mathsf{TestPwd}, \mathsf{sid}, P', \mathsf{pw}^*)$ to $\mathcal{F}_{\text{PAKE}}$.
   (2) Sample $b \leftarrow \mathcal{R}$ and compute $(K')^* := \mathsf{key}_2(b, A^*)$.
   (3) Send $(\mathsf{NewKey}, \mathsf{sid}, P', (K')^*)$ to $\mathcal{F}_{\text{PAKE}}$.

On $(\mathsf{sid}, \phi'^*)$ from $\mathcal{A}$ to $P$:

10. If either (1) $\mathsf{pw}^* = \bot \wedge \phi'^* = \phi'$ or (2) $\mathsf{pw}^* \neq \bot \wedge \phi'^* = \phi'$ and the password guess on $\mathsf{Deliver}$ was incorrect, send $(\mathsf{NewKey}, \mathsf{sid}, P, 0^\kappa)$ to $\mathcal{F}_{\text{PAKE}}$.
11. If $\mathsf{pw}^* \neq \bot \wedge \phi'^* = \phi'$ and the password guess on $\mathsf{Deliver}$ was correct:
    (1) Send $(\mathsf{TestPwd}, \mathsf{sid}, P, \mathsf{pw}^*)$ to $\mathcal{F}_{\text{PAKE}}$.
    (2) Compute $K^* = \mathsf{key}_1(H(\mathsf{pw}^*), \mathsf{msg}_2(b, A^*))$ and send $(\mathsf{NewKey}, \mathsf{sid}, P, K^*)$ to $\mathcal{F}_{\text{PAKE}}$.
12. Otherwise (i.e., if $\phi'^* \neq \phi'$ or $\phi'$ is undefined because no $\mathsf{Deliver}$ message has been sent):
    (1) Send $(\mathsf{Extract}, \mathsf{sid}, \phi'^*)$ from $\mathcal{F}_{\text{POPF}}$ to $\mathcal{A}$.
    (2) On $(\mathsf{Extract}, \mathsf{sid}, \mathsf{pw}'^*, \alpha^*)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{POPF}}$, send $(\mathsf{Eval}, \mathsf{sid}, \phi'^*, \mathsf{pw}'^*)$ from $\mathcal{F}_{\text{POPF}}$ to $\mathcal{A}$.
    (3) On $(\mathsf{Eval}, \mathsf{sid}, B^*)$ from $\mathcal{A}$ to $\mathcal{F}_{\text{POPF}}$,
        (i) If $\mathsf{pw}^* = \bot \wedge (\phi')^* \neq \phi'$, and there has been an $(\mathsf{Eval}, \mathsf{sid}, \phi', (\mathsf{pw}')^*)$ command to $\mathcal{F}_{\text{POPF}}$ whose answer is $(\mathsf{Eval}, \mathsf{sid}, B^*)$, then send $(\mathsf{TestSamePwd}, \mathsf{sid}, P, (\mathsf{pw}')^*)$ to $\mathcal{F}_{\text{PAKE}}$, compute $K^* := \mathsf{key}_1(H((\mathsf{pw}')^*), B^*)$, and send $(\mathsf{NewKey}, \mathsf{sid}, P, K^*)$ to $\mathcal{F}_{\text{PAKE}}$.
        (ii) Otherwise send $(\mathsf{TestPwd}, \mathsf{sid}, P, (\mathsf{pw}')^*)$ to $\mathcal{F}_{\text{PAKE}}$, compute $K^* := \mathsf{key}_1(H((\mathsf{pw}')^*), B^*)$, and send $(\mathsf{NewKey}, \mathsf{sid}, P, K^*)$ to $\mathcal{F}_{\text{PAKE}}$.

Simulation of $\mathcal{F}_{\text{POPF}}$: run the code of $\mathcal{F}_{\text{POPF}}$ as in (Figure 10) except that on $(\mathsf{Eval}, \mathsf{sid}, \phi', \mathsf{pw}^*)$, if the password guess on $\mathsf{Deliver}$ was correct return $B = \mathsf{msg}_2(b, A^*)$.

Figure 26: Simulator $\mathcal{S}$ for the EKE protocol realizing $\mathcal{F}_{\text{PAKE-sp}}$.

the key point is that in our setting *both before the hybrid and after the hybrid, the session keys of $P$ and $P'$ are the same*, so this hybrid is not needed anymore.

- Hybrid 9 in the previous proof — which changes $K$ from $K'$ (which is uniformly random since hybrid 2) back to the "real" $\mathsf{key}_1(a, B)$ in the re-encryption case — should change *both* $K$ and $K'$ back to $\mathsf{key}_1(a, B)$, because here $K$ and $K'$ need to be the same. This hybrid goes through due to KA security. □

## D.2 EKE Using IC Realizes $\mathcal{F}_{\text{PAKE}}$

We now turn to protocol EKE using IC, which is the EKE protocol in Appx. D.1 except that POPF Program is replaced by IC encryption and POPF Eval is replaced by IC decryption. we have:

**Theorem D.2.** *The EKE using IC protocol realizes $\mathcal{F}_{\text{PAKE}}$ in the $(\mathcal{F}_{\text{IC}}, \mathcal{F}_{\text{PAKE-1r}})$-hybrid world.*

*Proof (sketch).* This is simpler than Thm. D.1 so we only provide a more high-level sketch. The only modifications from the EKE-PRF simulator in Figure 20 are:

- In a NewKey command, we replace $\mathsf{PRF}_{K^*}(\phi')$ with $K^*$;
- We simulate the interface of $\mathcal{F}_{\text{IC}}$ rather than $\mathcal{F}_{\text{POPF}}$.

The key point is that now *the re-encryption case becomes non-existent*, since it is impossible to have $\phi' = \mathcal{E}(\mathsf{pw}, B)$, $(\phi')^* = \mathcal{E}(\mathsf{pw}, B)$, and $(\phi')^* \neq \phi'$. So we simply remove hybrids 8 and 9, and (as in the proof of Thm. D.1) also remove hybrids 5, 6, and 7. □

**Remark D.3.** *Our $\mathcal{F}_{\text{PAKE-sp}}$ functionality accurately captures the real adversary's ability while attacking EKE instantiated with POPF. The only additional attack compared to EKE using IC is re-encryption, where the adversary only attacks the $P$ session but can simultaneously check the password of $P'$* if the password guess for $P$ is correct *(in which case $P$ and $P'$ output the same key if and only if the passwords of $P$ and $P'$ match). This "conditional same password check" is exactly what the* TestSamePwd *command does.*