CAKE requires programming - On the provable post-quantum security of (O)CAKE

Kathrin Hövelmanns[1], Andreas Hülsing[1,2], Mikhail Kudinov[1], and Silvia Ritsch[1]

[1] Eindhoven University of Technology, The Netherlands
[2] SandboxAQ, Paolo Alto, US

Abstract. In this work we revisit the post-quantum security of KEM-based password-authenticated key exchange (PAKE), specifically that of (O)CAKE. So far, these schemes evaded a security proof considering quantum adversaries. We give a detailed analysis of why this is the case, determining the missing proof techniques. To this end, we first provide a proof of security in the post-quantum setting, up to a single gap. This proof already turns out to be technically involved, requiring advanced techniques to reason in the QROM, including the compressed oracle and the extractable QROM. To pave the way towards closing the gap, we then further identify an efficient simulator for the ideal cipher. This provides certain programming abilities as a necessary and sufficient condition to close the gap in the proof: we demonstrate that we can close the gap using the simulator, and give a meta-reduction based on KEM-anonymity that shows the impossibility of a non-programming reduction that covers a class of KEMs that includes Kyber / ML-KEM.

Keywords: Post-quantum cryptography, password-based authenticated key exchange, PAKE, CAKE, OCAKE, quantum ideal-cipher model, QIC, QROM, meta reduction.

1 Introduction

Passwords are still one of the most common means of user authentication. The reason is that they are easy to use on different devices, without the requirement for additional hardware. At the same time, passwords are a common reason for security breaches: They generally have low entropy, making them comparatively easy to guess, and as they have to be transmitted, they can potentially be intercepted. A way to avoid issues caused by interception is password-authenticated key exchange (PAKE). In a PAKE, the password is not transmitted but is used by communicating parties in a protocol to derive a shared session key. PAKE can not only be used for authentication on the Internet. It is a general tool for session authentication when only a low entropy secret is shared. This is why among the most popular applications of PAKE there is not only authentication in WiFi networks, but also authentication of contact-less connections to identity documents. For the latter use-case, the (low-entropy) machine-readable section of the document is read and used as a password in a PAKE (specifically PACE for ICAO passports).

This motivated major efforts put on the design of PAKE in recent years, including a competition within IETF's Crypto-Forum Research Group (CFRG) to select new PAKE. However, most of these designs are inherently linked to group-based cryptography, as they exploit its rich algebraic structure.

K.H. was supported by an NWO VENI grant (Project No. VI.Veni.222.397). A.H. and M.K. were supported by an NWO VIDI grant (Project No. VI.Vidi.193.066). S.R. is part of the Quantum-Safe Internet (QSI) ITN which received funding from the European Union's Horizon-Europe programme as Marie Sklodowska-Curie Action (PROJECT 101072637 - HORIZON -MSCA-2021-DN-01). The authors would like to thank Manuel Barbosa, Anne Broadbent, Christian Majenz, and Marc Vostermanns for helpful discussions.

This renders them vulnerable to attacks by future quantum computers. While this is not as big of an issue when just thinking about the authentication part, as PAKE are generally used in short-lived authentication decisions, it still is an issue in two dimensions. First, as PAKE are used to establish session keys, also the data later secured via these keys is susceptible to store-now-decrypt-later attacks. Second, given their use in long-lived identity documents and the specification and standardization time of such, even if we start transitioning now, we will have documents with old schemes in the field for the next 15-20 years. Hence, it is necessary to start looking for alternatives to classically secure PAKE now.

Indeed, there have been several works on the topic of post-quantum PAKE. In general, these can be split into two categories: First, there are proposals that build PAKE directly from specific hardness assumptions [DAL+17, SA23, DKBY24, TY19, SH19]. Second, there are generic proposals [BCP+23, PZ23] that can construct PAKE from generic Key Encapsulation Mechanisms (KEMs) which fulfill certain somewhat-standard properties ([Xag22, GMP22, MX23]). The former are mostly based on specific hardness assumptions on lattices and some come with a security proof. Some even come with a security proof that considers quantum adversaries[LLH24]. These proposals are usually quite efficient, as they can exploit the available algebraic structure. The latter do not have access to this structure, which can render them a bit less efficient. However, they have the advantage that they can be instantiated with a wide range of KEM. Given the experience that quantum computers are able to break virtually all traditional public key cryptography due their use of very few, closely related hardness assumptions, this freedom of choosing an arbitrary KEM, independent of the underlying hardness assumption, seems like a strong advantage in practice. For that reason, we study this latter category in this work.

The main contender of this category are the recently proposed CAKE & OCAKE protocols [BCP+23]. These are instances of the well-known Encrypted Key Exchange (EKE) blueprint that was also used in PACE, SPEKE, and many more. It has been demonstrated, in the UC-model [BCP+23] as well as in the game-based setting [PZ23, AHHR24], that CAKE & OCAKE are secure against attacks using classical computers as long as the used KEM provides three properties: It has to achieve indistinguishable keys under weak attacks (IND-CPA or IND-PCA); it has to guarantee anonymity of ciphertexts (ANO-CPA or ANO-PCA), meaning one cannot infer the used public key when seeing a ciphertext; and finally it has to guarantee that public keys are indistinguishable from random bit strings. However, a proof of security against quantum attackers is so far still missing.

KEM-EKE and its security. (O)CAKE is essentially a version of EKE instantiated using a KEM. We here consider OCAKE. CAKE only differs in minor aspects. The initiator **I** and the responder **R** share a password pw. The initiator generates an ephemeral KEM key pair pk, sk. Then it uses the password to compute an encrypted public key $apk \leftarrow \mathsf{BC}(\mathtt{KDF}(pw), pk)$ using a block cipher BC and sends apk to the responder. The responder recovers $pk \leftarrow \mathsf{BC}^{-1}(\mathtt{KDF}(pw), apk)$ using the password, and creates an encapsulation with it $(c, k) \leftarrow \mathsf{Encap}(pk)$. It sends c, potentially accompanied by a key-confirmation tag, back to **I** which can obtained the shared key $k \leftarrow \mathsf{Decap}(sk, c)$ using the ephemeral secret key. This is potentially followed by another key confirmation message from **I** to **R**.

A key property of PAKE is that a passive attack must not admit offline password-guessing attacks, and an active attack only allows to verify at most a single password guess. This is necessary due to the commonly limited size of practically used password spaces. For OCAKE this means – among others – that the second message must be computationally independent of the first. This is where KEM anonymity is required. Consider the following attack: A malicious initiator $\mathcal{A}$ sends a random apk to an honest **R**. In addition, $\mathcal{A}$ computes $pk_i \leftarrow \mathsf{BC}^{-1}(\mathtt{KDF}(pw_i), apk)$ for all passwords pw_i in the password space. This is possible due to the limited size of the password space. When receiving back the ciphertext c, the adversary simply checks under which pk_i the ciphertext c is

valid and extracts the respective password this way. If the KEM provides no anonymity, this attack is possible[3].

Consequently, if we want to base security on a computationally anonymous KEM, a reduction has to be able to use this very adversary to solve an anonymity challenge. In the formal definition of anonymity we are given two public keys and a ciphertext that is an encapsulation under one of the two. The task is to decide which of the two public keys was used to create the ciphertext. In the classical proof, this is done by simulating the ideal cipher via lazy sampling and adaptively programming. The latter is done in such a way that we are guaranteed that an *apk* which $\mathcal{A}$ eventually sends in an active attack decrypts to one of the challenge public keys for the anonymity game under the correct password with high probability. Then we use the challenge ciphertext as the second message. If $\mathcal{A}$ succeeds, we know that c was likely generated with the programmed public key. If $\mathcal{A}$ fails, c was likely generated using the other public key.

The quantum proof gap. When moving to a setting in which quantum adversaries are considered, one has to give such adversaries quantum access to random oracles and ideal ciphers as first observed in [BDF+11]. This makes a lot of proof steps involving random oracles that previously were simple standard arguments significantly more involved. Examples are adaptive reprogramming, online-extraction, or simple query-inspection. However, while challenging we have learned how to deal with these for random oracles. When it comes to ideal ciphers the situation is different. So far it is not known how to efficiently simulate an ideal cipher in a way that allows to make use of the above techniques.

This combination of challenging proofs and entirely missing techniques is what many expect to be the reason for OCAKE / CAKE not having a proof against quantum adversaries, yet. In this work we set out to understand what can actually be done and what is it that we are exactly lacking. Moreover, as researchers try to circumvent the ideal cipher model we are interested in what are possible venues and what paths are hopeless.

Our contribution In this work, we determine the necessary & sufficient (missing) ingredient for a reductionist proof to succeed, when considering quantum adversaries. Towards this end, we provide a proof against quantum adversaries in the quantum-accessible random oracle and ideal cipher model (QROM & QIC), assuming the existence of an efficient simulator for the ideal cipher that is capable of a certain form of reprogramming. We give a precise definition of this simulator and also demonstrate necessity of such a simulator when constructing an EKE PAKE from a KEM, basing security on computational anonymity (in contrast to statistical anonymity). This is done via a meta-reduction which shows that any reduction for anonymity that does not program the ideal cipher must be able to break anonymity itself. We show that the meta-reduction holds for KEM following the Fujisaki-Okamoto (FO) transform (or more precisely using the T-transform [HHK17]) which includes the new NIST standard ML-KEM, also known as Kyber, and most other post-quantum KEM considered in the NIST competition. Since OCAKE can only achieve post-quantum security when instantiated with a post-quantum KEM, we believe this to be the most relevant case.

Feasibility of instantiating the simulation. There has been some progress on simulations for random permutations that can support query-recording [Unr23], but to the best of our knowledge there is currently none that can fulfill the programmability requirement we isolate. In fact, it may prove impossible to achieve for an ideal cipher. While we phrase our results in most places with respect to an ideal cipher for ease of exposition, they can be easily generalized to families of bijections where each bijection is sampled from an independent distribution over the set of bijections.

Taking this generality into account, the programming requirement outlined in this paper immediately removes some constructions that try to avoid the QIC from consideration. One such example is a masking-based approach where the password (and potentially a session identifier) is hashed and

[3] Indeed, this attack is a slight simplification of the actual attack which is more complex due to the way anonymity is defined.

used as an xor-*mask* for the public key $apk = pk \oplus \mathtt{H}(pw, sid)$.[4] Since many efficient instantiations are ruled out by the requirement, the results of this paper indicate that it is worth investigating more complex instantiations that may come with a higher cost. We hope that our result can enable research into possible instantiations of the programming simulator we define, thereby leading the way to generic post-quantum PAKE from KEM with security against quantum adversaries.

Organization. The organization of this paper is depicted in Fig. 1

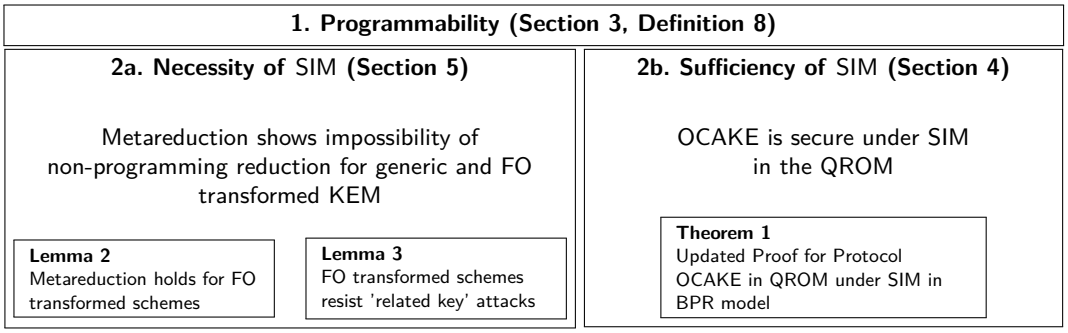


Fig. 1. Organization of this paper. We introduce the programmability requirement by defining a simulator in Section 3. We prove that programming is necessary using a meta-reduction technique in Section 5. To this end, we show that the meta-reduction holds for key encapsulation mechanisms that follow the Fujisaki-Okamoto (FO) transform. Finally, we show that the simulator is sufficient by giving an updated proof of the security of the OCAKE protocol in the quantum random oracle model (QROM) and the Bellare-Pointcheval-Rogaway (BPR) model in Section 4.

2 Preliminaries

In this section, we recall some notation and/or security notions for block ciphers, key encapsulation mechanisms, and hash functions, as well as the CAKE/OCAKE protocol.

Notation. We denote deterministic output y of an algorithm A on input x by $y := A(x)$. We denote algorithms with access to an oracle $\mathtt{O}$ by $A^{\mathtt{O}}$. Unless stated otherwise, we assume all our algorithms to be probabilistic and denote the computation by $y \leftarrow\$ A(x)$.

Block Ciphers The proof of security of the OCAKE protocol is given in the ideal cipher model [Bla06]. Analogous to the ROM for hash functions, the ideal cipher (IC) is an idealized description of a block cipher.

Definition 1 (Block Cipher (BC)). *A block cipher of block length n and key length k consists of two algorithms $\mathsf{BC} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ and $\mathsf{BC}^{-1} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ such that for every plaintext $m \in \{0,1\}^n$ and key $k \in \{0,1\}^k$, decryption undoes encryption: $\mathsf{BC}^{-1}(k, \mathsf{BC}(k, m)) = m$.*

Definition 2 (Ideal Cipher (IC)). *An ideal cipher is a collection of random permutations indexed by a key, to which all parties (including the adversary) are given oracle access. I.e., it is a pair of*

[4] To the curious reader: this masking cannot fulfill the programmability requirement due to malleability of the masking. Since the mask is not dependent on the public key, the random oracle cannot predict the adversary's choice of *apk* or *pk* from a query while the adversary learns the whole bijection from a single query.

random functions $\mathsf{E}, \mathsf{D} : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$*, such that* $\mathsf{D}(k, \mathsf{E}(k, m)) = m$ *and* $\mathsf{E}(k, \mathsf{D}(k, m)) = m$ *for all* k, m *in* $\mathcal{K} \times \mathcal{M}$*.*

Key Encapsulation Mechanisms (KEM). We start with the functional definition of KEMs. Afterwards we discuss their security.

Definition 3 (Key Encapsulation Mechanisms (KEMs)). *A* KEM *is a triple of algorithms* $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encap}, \mathsf{Decap})$*, together with a public key space* $\mathcal{PK}$ *and secret key space* $\mathcal{SK}$*.*

- $\mathsf{KGen} \to (pk, sk)$*: On empty input probabilistically* ***return*** *key pair* (pk, sk)*, where* pk *also defines a finite key space* $\mathcal{K}$ *and a ciphertext space* $\mathcal{C}$*.*
- $\mathsf{Encap}\ (pk) \to (c, K)$*: On input* pk *probabilistically* ***return*** *a pair* $(K, c) \in \mathcal{K} \times \mathcal{C}$*. We call* c *the* encapsulation *of the key* K*.*
- $\mathsf{Decap}\ (sk, c) \to K$*: On input* sk *and ciphertext* c *deterministically* ***return*** *a key* $K \in \mathcal{K}$*.*

Definition 4 ($\delta-$Correctness (average-case)). *We say that* KEM *is average-case* $(1-\delta)$*-correct if* $\Pr[\mathsf{Decap}(sk, c) = K | (c, K) \leftarrow\!\$\ \mathsf{Encap}(pk)] \geq 1 - \delta$*, where the probability is taken over* $(pk, sk) \leftarrow \mathsf{KGen}()$ *and the random coins of* Encap*.*

Definition 5 (γ-spreadness). *We say that PKE is* γ*-spread iff for all key pairs* $(pk, sk) \in supp(\mathsf{KGen})$ *and all messages* $m \in \mathcal{M}$ *it holds that*

$$\max_{c \in \mathcal{C}} \Pr[\mathsf{PKE.Enc}(pk, m) = c] \leq 2^{-\gamma},$$

where the probability is taken over the internal randomness of $\mathsf{PKE.Enc}$*.*

Definition 6 (Security notions for KEM). *Let* KEM *be a key encapsulation mechanism with public-key space* $\mathcal{PK}$ *and key space* $\mathcal{K}$*. We define the* $\mathsf{ANO\text{-}PCA}$ *game and the* $\mathsf{SROB\text{-}CPA}$ *game as depicted in Fig. 2, each relative to challenge bit* b*, and the respective advantage function of an adversary* $\mathcal{A}$ *against* KEM *as*

$$\mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{A}) := |\Pr[\mathsf{ANO\text{-}PCA}^0(\mathcal{A}) \Rightarrow 0] - \Pr[\mathsf{ANO\text{-}PCA}^1(\mathcal{A}) \Rightarrow 0]| \text{ and}$$
$$\mathbf{Adv}^{\mathsf{SROB\text{-}CPA}}_{\mathsf{KEM}}(\mathcal{A}) := |\Pr[\mathsf{SROB\text{-}CPA}^0(\mathcal{A}) \Rightarrow 0] - \Pr[\mathsf{SROB\text{-}CPA}^1(\mathcal{A}) \Rightarrow 0]|$$

Game $\mathsf{ANO\text{-}PCA}^b(\mathcal{A})$
01 $(pk_0, sk_0) \leftarrow\!\$\ \mathsf{KGen}$
02 $(pk_1, sk_1) \leftarrow\!\$\ \mathsf{KGen}$
03 $(c, K) \leftarrow \mathsf{Encap}(pk_0)$
04 $b' \leftarrow \mathcal{A}^{\text{1-PCO}}(pk_0, pk_1, c)$
05 **return** $[\![b = b']\!]$

Game $\mathsf{SROB\text{-}CPA}$
06 $(pk_0, sk_0) \leftarrow\!\$\ \mathsf{KGen}$
07 $(pk_1, sk_1) \leftarrow\!\$\ \mathsf{KGen}$
08 $c \leftarrow \mathcal{A}(pk_0, pk_1)$
09 $k_0 \leftarrow \mathsf{Decap}(pk_0, sk_0, c)$
10 $k_1 \leftarrow \mathsf{Decap}(pk_1, sk_1, c)$
11 return $[\![k_0 \neq \perp \wedge k_1 \neq \perp]\!]$

Fig. 2. The security games for anonymity ($\mathsf{ANO\text{-}PCA}$) and strong robustness ($\mathsf{SROB\text{-}CPA}$) for KEM. Oracle 1-PCO can be queried at most once.

Definition 7 (Multi-user security notions for KEM). *Let* KEM *be a key encapsulation mechanism with public-key space* $\mathcal{PK}$ *and key space* $\mathcal{K}$*. For integers* n *and* q_C*, we define the* PKU_n *game, the* $\mathsf{IND\text{-}CPA}_{n,q_C}$ *game and the* $\mathsf{ANO\text{-}PCA}_{n,q_C}$ *game as in Figures Fig. 3 and 3, each relative to challenge bit* b*, and the respective advantage function of an adversary* $\mathcal{A}$ *against* KEM *as*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CPA}_{n,q_C}}_{\mathsf{KEM}}(\mathcal{A}) := |\Pr[\mathsf{IND\text{-}CPA}^0_{n,q_C}(\mathcal{A}) \Rightarrow 0] - \Pr[\mathsf{IND\text{-}CPA}^1_{n,q_C}(\mathcal{A}) \Rightarrow 0]| \text{ and}$$
$$\mathbf{Adv}^{\mathsf{ANO\text{-}PCA}_{n,q_C}}_{\mathsf{KEM}}(\mathcal{A}) := |\Pr[\mathsf{ANO\text{-}PCA}^0_{n,q_C}(\mathcal{A}) \Rightarrow 0] - \Pr[\mathsf{ANO\text{-}PCA}^1_{n,q_C}(\mathcal{A}) \Rightarrow 0]|$$

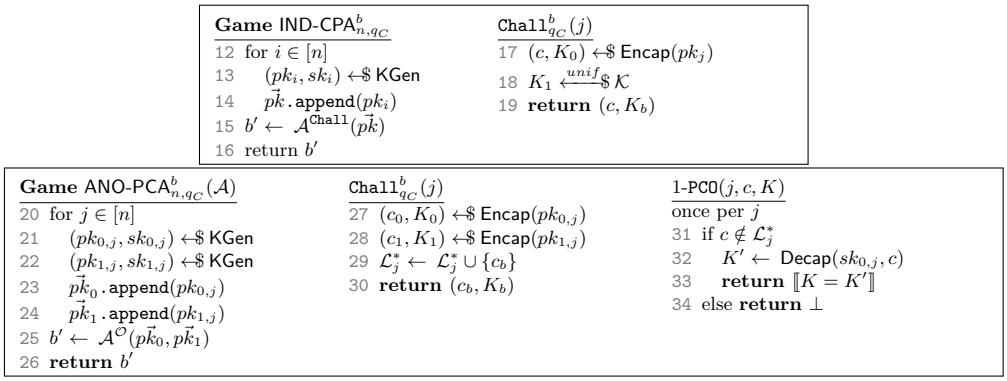


Fig. 3. The Multi-user games for anonymity (ANO-PCA$_{n,q_C}$) and indistinguishability (IND-CPA$_{n,q_C}$) for KEM, for n users. By $\mathcal{O}$ we denote $\mathcal{A}$'s oracles $\mathcal{O} = \{\mathtt{1\text{-}PCO}, \mathtt{Chall}^b_{q_C}\}$. Challenge oracle Chall can be queried at most q_C many times per user, and 1-PCO once per public key.

Extractable Compressed Random Oracle. Here we introduce an extension of Zhandry's Compressed Random Oracle [Zha19] called extractable Compressed Random Oracle [DFMS22] (eCO). Roughly speaking using eCO represents a quantum-accessible Random Oracle with an additional mechanism that allows to check if there was a query that satisfies some function f. This gives us a form of observable QROM. Lets discuss it in more details.

In [DFMS22] eCO was developed. eCO has two interfaces: one that mimics the behavior of a random oracle (eCO.RO) and another that allows extraction of information (eCO.E). The $\mathtt{eCO.RO}_f$ interface defined relative to a function f takes a classical value t as input and simulates a quantum measurement that "collapses" the oracle database, allowing it to yield a specific outcome x. After the measurement, the database is in a state where all the values for eCO.RO collapse to the values y that satisfy the equation $f(x, y) = t$. Also no earlier entries x' will have any possibilities y' left that also satisfy the equation $f(x', y') = t$. When the function f used is clear from context we write eCO.E instead of $\mathtt{eCO.E}_f$. To clarify, consider the classical analogue where the Random Oracle is simulated via lazy-sampling. Then for an extraction query we go through the database and output the first pair (x, y) such that x is the smallest value for which $f(x, y) = t$.

Previously, in [DFMS22, HM24] the eCO was used to simulate Decapsulation queries during the runtime of the adversary. In our case we will need it to check if the adversary was able to check if adversarially generated tags are valid.

Now lets give a more formal treatment of the eCO. We will closely follow the description from [DFMS22, HM24]. The description represents an inefficient version of the eCO which can be made efficient using sparse encoding (c.f., [DFMS22]). The simulator eCO for a random function O: $\{0,1\}^m \to \{0,1\}^n$ is a stateful oracle with a state stored in a quantum register $D = D_{0^m} \dots D_{1^m}$, where for each input value $x \in \{0,1\}^m$, register D_x has $n+1$ qubits used to store superpositions of n-bit output strings y, encoded as $0y$, and an additional symbol $\perp$, encoded as 10^n. We adopt the convention that an operator expecting n input qubits acts on the last n qubits when applied to one of the registers D_x. The compressed oracle has the following three components.

1. The initial state of the oracle, $|\phi\rangle = |\perp\rangle^{2^m}$
2. A quantum query with query input register X and output register Y is answered using the oracle unitary O_{XYD} defined by

$$O_{XYD} |X\rangle_X = |X\rangle_X \otimes \left(F_{D_x} \mathrm{CNOT}^{\otimes n}_{D_x:Y} F_{D_x}\right),$$

where $F|\bot\rangle = |\phi_0\rangle, F|\phi_0\rangle = |\bot\rangle$ and $F|\psi\rangle = |\psi\rangle$ for all $|\psi\rangle$ such that $\langle\psi|\bot\rangle = \langle\psi|\phi_0\rangle = 0$, with $|\phi_0\rangle = |+\rangle^{\otimes n}$ being the uniform superposition.
3. A recovery algorithm that recovers a standard QRO O: apply $F^{\otimes 2^m}$ to D and measure it to obtain the function table of O.

The formal description of the extraction interface is the following. Given a random oracle $O : \{0,1\}^m \to \{0,1\}^n$, let $f : \{0,1\}^m \times \{0,1\}^n \to \{0,1\}^\ell$ be a function. We define a family of measurements $(\mathcal{M}^t)_{t\in\{0,1\}^\ell}$. The measurement $\mathcal{M}^t$ has measurement projectors $\{\Sigma^{t,x}\}_{x\in\{0,1\}^m\cup\{\emptyset\}}$ defined as follows. For $x \in \{0,1\}^m$, the projector selects the case where D_x is the first (in lexicographical order) register that contains y such that $f(x,y) = t$, i.e.

$$\Sigma^{t,x} = \bigotimes_{x'<x} \bar{\Pi}^{t,x'}_{D'_x} \otimes \Pi^{t,x}_{D_x}, \quad \text{with} \quad \Pi^{t,x} = \sum_{\substack{y\in\{0,1\}^n:\\ f(x,y)=t}} |y\rangle\langle y|$$

and $\bar{\Pi} = \mathbb{1} - \Pi$. The remaining projector corresponds to the case where no register contains such a y, i.e.

$$\Sigma^{t,\emptyset} = \bigotimes_{x'\in\{0,1\}^m} \bar{\Pi}^{t,x'}_{D'_x}$$

eCO is initialized with the initial state of the compressed oracle. eCO.RO is quantum-accessible, eCO.E is a classical oracle interface that, on input t, applies $\mathcal{M}^t$ to eCO's internal state. The following properties of eCO are required for the main lemma that will be used in our proof. We define a relationship R based on function f and a target value t as all input output pairs that satisfy function $f(x,y) = t$.

$$R_{f,t}(x,y) :\Leftrightarrow f(x,y) = t.$$

Next, lets define

$$\Gamma_R := \max_x |\{y \mid R(x,y)\}|.$$

Γ_R outputs the maximum number values y that satisfy a relationship R for some input x. And finally we define

$$\Gamma(f) = \max_t \Gamma_{R_{f,t}}.$$

$\Gamma(f)$ outputs the maximum number of outputs y that satisfy some relationship $R_{f,t}$ (maximizing over target values t) for some input x. The $\Gamma(f)$ value determines whether we are able to use extractions without disturbing the random oracle too much. This is stated in the following lemma.

Lemma 1 (Part of theorem 3.4 in [DFMS22], formulated in [HM24]). *The extractable RO simulator eCO described above, with interfaces eCO.RO and eCO.E, satisfies the following properties.*

1. *If eCO.E is unused, eCO is perfectly indistinguishable from a random oracle.*
2. *Any two subsequent independent queries to eCO.RO commute. In particular, two subsequent classical eCO.RO-queries with the same input x give identical responses.*
3. *Any two subsequent independent queries to eCO.E commute. In particular, two subsequent eCO.E queries with the same input t give identical responses.*
4. *Any two subsequent independent queries to eCO.E and eCO.RO $8\sqrt{2\Gamma(f)/2^n}$-almost-commute.*

In our application we will not give the adversary the extraction interface. We still restrict the adversary only to the RO queries. On the other hand the reduction is able to perform extraction queries to check if the oracle has been queried on the inputs that are interesting for us.

Transformation T [HHK17] T transforms an OW-CPA secure public-key encryption scheme into an OW-PCA secure one.

To a public-key encryption scheme $\mathsf{PKE} = (\mathsf{PKE.KGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ with message space $\mathcal{M}$ and randomness space $\mathcal{R}$, and random oracle $\mathtt{G}{:}\mathcal{M} \to \mathcal{R}$, we associate $\mathsf{PKE}_1 = \mathbf{T}[\mathsf{PKE}, \mathtt{G}\,]$. The algorithms of $\mathsf{PKE}_1 = (\mathsf{PKE.KGen}, \mathsf{PKE.Enc}_1, \mathsf{PKE.Dec}_1)$ are defined in Fig. 4. Note that Enc1 deterministically computers the ciphertext as $c := \mathsf{PKE.Enc}(pk, m; \mathtt{G}(m))$.

$\underline{\mathsf{PKE.Enc}_1(pk, m)}$	$\underline{\mathsf{PKE.Dec}_1(sk, c)}$
35 $c := \mathsf{PKE.Enc}(pk, m; \mathtt{G}(m))$	37 $m' := \mathsf{PKE.Dec}(sk, c)$.
36 return c	38 if $m' = \bot$ or $\mathsf{PKE.Enc}(pk, m'; \mathtt{G}(m')) \neq c$
	39 return $\bot$
	40 else return m'

Fig. 4. The algorithms of the T-Transform.

2.1 Protocols CAKE and OCAKE

In our analysis of which proof techniques are required for post-quantum PAKE, we focus on two related protocols: CAKE and OCAKE, as shown in Fig. 5. In both protocols, two parties in possession of a shared password pw want to establish a shared session key. First, the initiator **I** will generate a KEM key pair, encrypt the public key pk with a symmetric cipher, and send the encrypted public key apk to the responder **R**. Upon receiving apk, **R** uses the password to recover pk, which it then used to compute an encapsulation c and a pre-key K. In the OCAKE protocol, **R** sends the encapsulation c to **I**, together with a responder tag tag that can be used by **I** to confirm the key. In the CAKE protocol, instead of sending c and the tag, **R** sends an encryption of c, again using a symmetric cipher. In both cases, **R** derives its session key SK' from pre-key K and the session transcript. Upon receiving (the encryption of) c, **I** (decrypts and) decapsulates the ciphertext to obtain a pre-key K', which can then again be used to derive the session key. In OCAKE, **I** only derives the session key if key confirmation succeeds, i.e., if the received tag equals the one **I** derives from its own state.

Assuming that both parties used the same password and that KEM worked correctly, **R** and **I** end up with the same session key. Both protocols require an additional round of key confirmation in order to achieve explicit authentication. However, as CAKE is implicitly authenticating for both parties, it requires confirmation for both parties, while OCAKE is explicitly rejecting on the initiator side (the Responder or server is explicitly authenticated.)

We made a minor conceptual modification to the previous descriptions of the OCAKE protocol: we derive tag and SK in the same call to $\mathtt{H}$. This variant constitutes a minor optimization and simplifies the analysis.

The first security proofs for these two protocols were given in [BCP+23] in the UC framework and only considered attackers without quantum capabilities. As a first step towards capturing security even against attackers with quantum capabilities, [PZ23] and [AHH+23] gave proofs in the BPR model for CAKE and OCAKE, respectively. The analysis in this work is based on these security proofs.

3 Programming Simulator SIM for (O)CAKE

We now define a simulation that is (computationally) indistinguishable from an ideal cipher, but answers queries by embedding values from a challenge set. This simulation models a block cipher that can be queried on quantum states.
We later prove that this simulator is sufficient to prove security against quantum adversaries of

Initiator I (Client)	**Protocol OCAKE**	**Responder R (Server)**
Password pw		Password pw
	Public Key	
$k_{pw} \leftarrow \mathtt{KDF}(pw)$		$k_{pw} \leftarrow \mathtt{KDF}(pw)$
$(pk, sk) \leftarrow\!\$\ \mathsf{KGen}$		
$apk \leftarrow \mathsf{BC}(k_{pw}, pk)$	$\xrightarrow{apk}$	$pk' \leftarrow \mathsf{BC}^{-1}(k_{pw}, apk)$
	Session Pre-Key	
$K' \leftarrow \mathsf{Decap}(sk, c)$	$\xleftarrow{c}$	$(c, K) \leftarrow\!\$\ \mathsf{Encap}(pk')$
	Key Derivation	
	$\xleftarrow{tag}$	$tag, SK \leftarrow \mathtt{H}(K, pw, apk, pk', c)$
$tag', SK' \leftarrow \mathtt{H}(K', pw, apk, pk, c)$		$\mathtt{output}\ SK$
$\mathtt{if}\ tag' = tag$		
$\quad \mathtt{output}\ SK'$		

Initiator I (Client)	**Protocol CAKE**	**Responder R (Server)**
Password pw		Password pw
	Public Key	
$(pk, sk) \leftarrow\!\$\ \mathsf{KGen}$		
$e_1 \leftarrow \mathsf{BC}_1(pw, pk)$	$\xrightarrow{e_1}$	$pk' \leftarrow \mathsf{BC}_1^{-1}(pw, e_1)$
	Session Pre-Key	
		$(c, K) \leftarrow\!\$\ \mathsf{Encap}(pk')$
$c' \leftarrow \mathsf{BC}_2^{-1}(pw, e_2)$	$\xleftarrow{e_2}$	$e_2 \leftarrow \mathsf{BC}_2(pw, c)$
$K' \leftarrow \mathsf{Decap}(sk, c')$		
	Key Derivation	
$SK' \leftarrow \mathtt{H}(pk, c', K', pw)$		$SK \leftarrow \mathtt{H}(pk', c, K, pw)$
$\mathtt{output}\ SK'$		$\mathtt{output}\ SK$

Fig. 5. (top) The OCAKE protocol, using a key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encap}, \mathsf{Decap})$, key derivation function KDF, hash function H and a block cipher $\mathsf{BC} = (\mathsf{BC}, \mathsf{BC}^{-1})$. This variant enables an optimization where tag and SK are computed using the same call to H. (bottom) The CAKE protocol, using a key encapsulation mechanism $\mathsf{KEM} = (\mathsf{KGen}, \mathsf{Encap}, \mathsf{Decap})$, hash function H and a block cipher $\mathsf{BC} = (\mathsf{BC}, \mathsf{BC}^{-1})$. In both cases, existing proofs model the block cipher as an ideal cipher.

OCAKE in the BPR model in Section 4 by giving such a proof. Lastly, we use a meta-reduction technique([FLR+10, FF13, BV98]) in section 5 to study how programmability as presented by SIM is necessary for any security proof of OCAKE.

The programming simulator. We now introduce the simulator that describes the minimal necessary requirements to prove security of OCAKE. Given the current uncertainty in achieving programmability with quantum queries, this definition aims to maximize flexibility in its instantiation while clearly outlining the requirements.
Consider an adversary $\mathcal{A}$, a message space $\mathcal{M}$, and a key space $\mathcal{K}_{\mathsf{SIM}}$. From this we define a family of bijections $\mathsf{E}_k : \mathcal{M} \to \mathcal{M}$ indexed on a key $k \in \mathcal{K}_{\mathsf{SIM}}$ where each bijection independently follows distribution $\mathcal{B}$. We denote the inverse of E_k as D_k. We also use $\mathsf{E}(k, m)$ to denote $\mathsf{E}_k(m)$.
In the case where $\mathcal{B}$ is the uniform distribution over all permutations, this definition is equivalent to that of an ideal cipher.

Definition 8 ($(\delta_{\mathsf{SIM}}, \varepsilon_{\mathsf{SIM}})-$Programming Simulator SIM (single-session)). *Let adversary $\mathcal{A}$, message space $\mathcal{M}$, key space $\mathcal{K}_{\mathsf{SIM}}$ and E be as above.We now define simulator SIM: ($\mathsf{E}_{\mathsf{SIM}}$,$\mathsf{D}_{\mathsf{SIM}}$) relative to a challenge set $\mathcal{X} \subset \mathcal{M}$ of non-repeating, independently and uniformly random elements of the message space $\mathcal{M}$. We say that SIM is a $(\delta_{\mathsf{SIM}}, \varepsilon_{\mathsf{SIM}})-$programming simulator whenever we have for any adversary $\mathcal{A}$ that interacts with the oracles and eventually outputs some value apk and a session ID that:*

1. SIM *is indistinguishable from family* E *up to some failure probability $\varepsilon_{\mathsf{SIM}}$, i.e., the success probability of any distinguishing adversary $\mathcal{D}$ that wins if it outputs* 0 *when interacting with* E, D *and* 1 *when interacting with* SIM *($\mathcal{X}$) is bounded by $\varepsilon_{\mathsf{SIM}}$:*

$$|\Pr[\mathcal{D}^{\mathsf{E},\mathsf{D}} \Rightarrow 0] - \Pr[\mathcal{D}^{\mathsf{SIM}(\mathcal{X})} \Rightarrow 0]| \leq \varepsilon_{\mathsf{SIM}}.$$

2. SIM *ensures that apk's decryption is in the challenge set $\mathcal{X}$ with probability $1 - \delta_{\mathsf{SIM}}$:*

$$\Pr[(apk, sid) \leftarrow \mathcal{A}^{\mathsf{SIM}(\mathcal{X})} : \mathsf{D}_{\mathsf{SIM}}(K[sid], apk) \in \mathcal{X}] = 1 - \delta.$$

where $K[sid]$ is the key associated with session sid.

Generalization to multiple sessions We define the multi-session variant of SIM analogously to the single-session version, except the adversary now outputs n_s pairs (apk_i, sid_i). Since in a reduction, it could be that the attacker selects any of these values to attack, it is necessary that all n decryptions are in the challenge set: $\forall(apk_i, sid_i)$ output by $\mathcal{A}$, we require that $\mathsf{D}_{k(sid_i)}(apk_i) \in \mathcal{X}$ with probability $1 - \delta_{\mathsf{SIM}}$.

Definition 9 ($(n, \delta_{\mathsf{SIM}}, \varepsilon_{\mathsf{SIM}})-$Programming Simulator SIM (multi-session)). *Let adversary $\mathcal{A}$, message space $\mathcal{M}$, key space $\mathcal{K}_{\mathsf{SIM}}$ and E be as above.We now define simulator SIM: ($\mathsf{E}_{\mathsf{SIM}}$,$\mathsf{D}_{\mathsf{SIM}}$) relative to a challenge set $\mathcal{X} \subset \mathcal{M}$ of non-repeating, independently and uniformly random elements of the message space $\mathcal{M}$. We say that SIM is a $(\delta_{\mathsf{SIM}}, \varepsilon_{\mathsf{SIM}})-$programming simulator whenever we have for any adversary $\mathcal{A}$ that interacts with the oracles and eventually outputs some values apk_i and a session IDs sid_i, $i \in \{0, 1, \ldots, n-1\}$ that:*

1. SIM *is indistinguishable from family* E *up to some failure probability $\varepsilon_{\mathsf{SIM}}$, i.e., the success probability of any distinguishing adversary $\mathcal{D}$ that wins if it outputs* 0 *when interacting with* E, D *and* 1 *when interacting with* SIM *($\mathcal{X}$) is bounded by $\varepsilon_{\mathsf{SIM}}$:*

$$|\Pr[\mathcal{D}^{\mathsf{E},\mathsf{D}} \Rightarrow 0] - \Pr[\mathcal{D}^{\mathsf{SIM}(\mathcal{X})} \Rightarrow 0]| \leq \varepsilon_{\mathsf{SIM}}.$$

2. SIM *ensures that the decryption of all* apk_i *is in the challenge set* $\mathcal{X}$ *with probability* $1 - \delta_{\mathsf{SIM}}$*:*

$$\Pr[(\vec{apk}, \vec{sid}) \leftarrow \mathcal{A}^{\mathsf{SIM}(\mathcal{X})} : \\ \forall (sid_i, apk_i) \in (\vec{sid}, \vec{apk}) : \mathsf{D}_{\mathsf{SIM}}(K[sid_i], apk_i) \in \mathcal{X}] = (1 - \delta_{\mathsf{SIM}}).$$

where $K[sid]$ *is the key associated with session* sid*.*

4 Security of (O)CAKE in the QROM

To prove security against quantum attackers, it is necessary to consider the quantum random oracle model as well as a quantum-accessible model for the symmetric cipher. We show that the simulation described in section 3 is sufficient to prove security of PAKE in the BPR model, using a generic KEM, by providing an updated security statement for OCAKE (Section 4.1). We also provide a high-level intuition for how to update the proof for CAKE (Section 4.2).

4.1 Security of OCAKE under SIM

Security of the OCAKE protocol has been shown in the universal composability framework [BCP+23] as well as in the BPR model [AHH+23]. Both results use the ROM and the ICM. (Appendix B gives a more thorough analysis of the role of the ideal cipher in the proof and its relation to attacks.) Building on the proof given in [AHH+23], we give a security proof for the OCAKE protocol in the BPR model, using the QROM and assuming the existence of a simulator SIM that effectively replaces the IC.

To lift the classical proof to the quantum setting, all *public* functions – that is, those that can be computed by the attacker – need to be modeled to allow quantum queries [BDF+11]. Therefore, any proof argument that considers hash functions, key derivation functions or block ciphers needs adapting to quantum queries. In this setting, we now give our new security theorem.

Theorem 1 (Tight security of OCAKE in the QROM from multi-user security of KEM and SIM). *Let* KEM *be a key encapsulation mechanism that is* $(1 - \delta^{\mathsf{KEM}}_{corr})$*-correct, let* KDF*, and* H *be modeled as quantum random oracles with domains* $\mathcal{K}_{pw}$ *and* $\mathcal{T} \times \mathcal{SK}$*,* BC *be a quantumly-accessible block cipher that we model according to* SIM*, and let* $\mathcal{A}$ *be a* BPR *adversary against* OCAKE[KEM, KDF, H, BC]*, issuing at most* n_a *many* ***Send*** *queries (active attacks),* n_p *many* ***Execute*** *queries (passive attacks), and* q_{RO} *many queries to its respective random oracles. Let* $n_s := n_a + n_p$ *be the total number of sessions. Then there exist a multi-user-*IND-CPA *adversary* $\mathcal{B}^{\mathsf{IND}}$ *and a multi-user-*ANO-PCA *adversary* $\mathcal{B}^{\mathsf{ANO}}$ *against* KEM *such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{BPR}}_{\mathsf{OCAKE}}(\mathcal{A}) \leq{}& \frac{n_a}{|\mathcal{D}|} + n_a^2 \cdot \eta_{\mathsf{KGen}} + \frac{|\mathcal{D}|^2}{|\mathcal{K}_{pw}|} + \frac{80e^2 q_H^3 + 2}{|\mathcal{T}|} + n_a \cdot \delta^{\mathsf{KEM}}_{corr} + \\ &+ 4 \cdot n_a \cdot (n_s + q_H)\sqrt{2/|\mathcal{T}|} + 2 \cdot (\frac{1}{(1 - \delta_{\mathsf{SIM}})} \cdot \mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}} + \varepsilon_{\mathsf{SIM}}) \\ &+ 2 \cdot n_s \cdot (n_a + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM}} + 2 \cdot \frac{q_H}{|\mathcal{K}|}\end{aligned}$$

and the running time of $\mathcal{B}^{\mathsf{IND}}$*,* $\mathcal{B}^{\mathsf{ANO}}$*, and* $\mathcal{B}^{\mathsf{PKU}}$ *is about that of* $\mathcal{A}$*.*

To obtain this bound, we update the classical BPR proof for OCAKE [AHH+23] to capturing quantum attackers. We give an overview over the classical proof's game-hops in Table Fig. 6, indicate which game-hops we need to update, the new proof techniques we use to do so, and the resulting loss.

To summarize our updates, firstly, we lift all game-hops that use random oracle properties (games $\mathbf{G}_5$, $\mathbf{G}_{13}$) to the QROM. Secondly, for a concise presentation of how to apply our simulator properties, we fold all game-hops that use ideal cipher properties (games $\mathbf{G}_3$,$\mathbf{G}_4$,$\mathbf{G}_6$,$\mathbf{G}_{10}$) into a single game ($\mathbf{G}_{10}$). The resulting upgraded proof is fully adapted to capturing quantum attackers.

We describe each game-hop in this section and give the intuition that underlies them. For the sake of completeness, we provide detailed reductions (including pseudocode) in Appendix A.2. For the reader's convenience, we also recall the details of the BPR security model in Appendix A.1.

Game	Description	Update?	Loss
$\mathbf{G}_0$	Original BPR game		
$\mathbf{G}_1$	Abort on KGen Collision.	Not necessary	$n_a^2 \cdot \eta_{\mathsf{KGen}}$
$\mathbf{G}_2$	Abort on KDF Collision.	Not necessary	$\frac{\lvert\mathcal{D}\rvert^2}{\lvert\mathcal{K}_{pw}\rvert}$
$\mathbf{G}_3$	IC lazy sampling w/ abort	This work (SIM)	-
$\mathbf{G}_4$	Prevent IC collisions	This work (SIM)	-
$\mathbf{G}_5$	Abort on Tag Collision.	This work (CO)	$\frac{80e^2 q_{\mathtt{H}}^3+2}{\lvert\mathcal{T}\rvert}$
$\mathbf{G}_7$	Do not decrypt honest c.	Not necessary	$n_a \cdot \delta_{corr}^{\mathsf{KEM}}$
$\mathbf{G}_6$	Sample IC using KGen	This work (SIM)	-
$\mathbf{G}_8$	Abort on tag under correct pw.	This work (eCO)	$\Pr[\,\mathsf{corrPW}\,]$
$\mathbf{G}_9$	Make eCO extractions online.	This work (eCO)	$n_a \cdot (n_s + q_{\mathtt{H}})\sqrt{2/\lvert\mathcal{T}\rvert}$
$\mathbf{G}_{10}$	Randomize public key.	This work (SIM)	$\frac{1}{(1-\delta_{\mathsf{SIM}})} \cdot \mathbf{Adv}_{\mathsf{KEM}}^{\text{ANO-PCA}} + \varepsilon_{\mathsf{SIM}}$
$\mathbf{G}_{11}$	Randomize pre-key.	Not necessary	$n_s \cdot (n_a+1) \cdot \mathbf{Adv}_{\mathsf{KEM}}^{\text{IND-CPA}}$
$\mathbf{G}_{12}$	Move eCO extractions to end.	This work (eCO)	$n_a \cdot (n_s + q_{\mathtt{H}})\sqrt{2/\lvert\mathcal{T}\rvert}$
$\mathbf{G}_{13}$	Randomize Tag, Session Key.	This work (CO)	$\frac{q_{\mathtt{H}}}{\lvert\mathcal{K}\rvert}$
$\mathbf{G}_{15}$	Randomize Passwords.	Not necessary	0

Fig. 6. Game-hops in the classical BPR proof for OCAKE. The game-hops that needed updating use new techniques: compressed oracles (CO), extractable compressed oracles (eCO), and the simulator defined in this work (SIM).

We will use the notational convention $\mathbf{Adv}_{\mathsf{i}} := |\Pr[\mathbf{G}_i(\mathcal{A}) \Rightarrow 1]|$.

Game $\mathbf{G}_0$: Original BPR *game. The* BPR *oracles are exactly as in the existing* BPR *proof (see App. A.1).* `H, KDF,KDF'` *and* IC *are quantum-accessible.*

Game $\mathbf{G}_1$: Handling collisions of KEM key pairs. (Unchanged) The game aborts whenever there are at least two sessions that sampled the same ephemeral KEM key pair (pk, sk). Let η_{KGen} be the collision probability of KGen. Since games $\mathbf{G}_0$ and $\mathbf{G}_1$ are identical unless a collision occurs, we have

$$|\mathbf{Adv}_0 - \mathbf{Adv}_1| = \Pr[\mathtt{KDFColl}] \leq n_a^2 \cdot \eta_{\mathsf{KGen}}.$$

Game $\mathbf{G}_2$: Abort on KDF Collision. The game aborts whenever two passwords lead to a collision in the key derivation function that maps passwords to block cipher keys. This bound only depends on the choice of KDF and dictionary $\mathcal{D}$, not on any adversarial input. The resulting bound thus is statistical. Concretely, if KDF is a uniformly random function $f : \mathcal{D} \to \mathcal{K}_{pw}$, the probability that two inputs pw, pw' are mapped to the same output k_{pw} can be upper-bounded by the birthday collision bound: $\mathbf{Adv}_2 \leq \Pr[\,\mathsf{KDFColl}\,] \leq \frac{|\mathcal{D}|^2}{|\mathcal{K}_{pw}|}$.

Games $\boldsymbol{G}_3$ *and* $\boldsymbol{G}_4$: we fold them into game $\mathbf{G}_{10}$.

Game $\boldsymbol{G}_5$*: Abort on Tag Collision.* One of the informal security goals of PAKE proofs is to rule out attacks where an adversary can *test* more than one password in a single online attack. One example of such an attack is the exploitation of a tag collision: assume there exist two passwords pw, pw' s.t. the responder tag *tag* computed from them for a given transcript is the same. Then the attacker could win the game by submitting *tag* without having uniquely identified the right password.
Therefore, $\mathbf{G}_5$ aborts whenever the attacker has found such a collision in $\mathtt{H}$. To argue about the distance between $\mathbf{G}_5$ and $\mathbf{G}_4$, we use the compressed oracle technique [CFHL21].
Let $\mathcal{A}$ be the adversary in the PAKE game. We define an adversary $\mathcal{B}$ in the collision-finding game that runs $\mathcal{A}$ and is successful if it outputs x, x' s.t. $\mathtt{H}(x) = \mathtt{H}(x')$ at the end of the game.
$\mathcal{B}$ simulates $\mathcal{A}$'s PAKE and random oracle queries. First, $\mathcal{B}$ replaces the random oracle with a compressed random oracle. The compressed random oracle allows $\mathcal{B}$ to record $\mathcal{A}$'s queries in a database.
Then, at the end of the game, $\mathcal{B}$ measures the database for collisions. Let R be the relation that describes a collision in the tag: for distinct x, x' and arbitrary sk, sk' we have that $\mathtt{H}(x)||sk = \mathtt{H}(x')||sk'$. Then we have $R \subseteq \mathcal{X}^l \times (\mathcal{T} \times \mathcal{SK})^l$ for $l = 2$.

$$\mathsf{CL} := \{D | \exists x \neq x' : D(x) = D(x') \neq \perp\}$$

Then, by Corollary 4.2 [CFHL21, Corollary 4.2], we have that for success probability in the original game p and success probability in the game using the compressed random oracle p', we have that $\sqrt{p} \leq \sqrt{p'} + \sqrt{\frac{2}{|\mathcal{T}|}}$.
We now argue about the success probability p', closely following the reasoning in section 2.3 of [CFHL21], (*Finding a Collision (with Parallel Queries)*). Let $\mathsf{SZ}_{\leq(s-1)}$ be a database of size at most $s-1$. We set $k = 1$ as we do not model parallel queries, so we have using the transition probability of a database that does not contain a collision to one that *does*

$$\sqrt{p'} \leq \sum_{s=1}^{q} [\![\mathsf{SZ}_{\leq(s-1)} \backslash \mathsf{CL} \to \mathsf{CL}]\!] \leq \sum_{s=1}^{q} 2e\sqrt{10\frac{q_{\mathtt{H}}}{|\mathcal{T}|}} \leq 2qe\sqrt{10\frac{q_{\mathtt{H}}}{|\mathcal{T}|}}$$

where e is Euler's number. Therefore we conclude $p' \leq \frac{40q^3e^2}{|\mathcal{T}|}$. From $\sqrt{p} \leq \sqrt{p'} + \sqrt{\frac{2}{|\mathcal{T}|}}$ we then see by squaring that

$$p \leq p' + \frac{2}{|\mathcal{T}|} + \sqrt{\frac{2p'}{|\mathcal{T}|}} \leq \frac{40e^2(q^3 + \sqrt{2}q^{3/2}) + 2}{|\mathcal{T}|} \leq \frac{80e^2q^3 + 2}{|\mathcal{T}|}$$

Finally, this allows us to bound the distance:

$$\mathbf{Adv}_5 \leq p \leq \frac{80e^2q_{\mathtt{H}}^3 + 2}{|\mathcal{T}|}.$$

Game $\boldsymbol{G}_6$ is also folded into game $\mathbf{G}_{10}$.

Game $\boldsymbol{G}_7$*: Correctness. (Unchanged)* In game $\mathbf{G}_7$, whenever there is a flow 2 query where the message was honestly generated by a matching session, we do not decapsulate to obtain the pre-key. Instead, if the message was generated by a matching session, we use the pre-key that was encapsulated by that instance. Adversarially generated messages as well as ones that are forwarded from a non-matching session are decapsulated as before. As in the classical proof, the distance to the previous game can be bounded by $\mathbf{Adv}_7 \leq n_a \cdot \delta_{corr}^{\mathsf{KEM}}$.

Game $\boldsymbol{G}_8$: Abort on dishonest tags using correct pw. We now abort whenever the adversary actively interacts with a session, meaning the session is not executed between two honest parties, and submits a tag that was generated using the correct password. This models the case that the attacker successfully guessed the password and used it to attack an honest part, which can happen both for

1. initiators, in which case the attacker sent a responder tag *tag* that uses the correct password, and for
2. responders, in which case the attacker sent an *apk* that uses the correct password.

Here we focus on case 1, since the other case is handled by simulation SIM (see 3). We need to adapt previous classical reasoning about this change to capture that oracle $\mathtt{H}$ now is quantum-accessible – previous reasoning exploited that game and reductions can observe the issued queries to $\mathtt{H}$, thus being able to immediately notice that *tag* was generated with the correct password. As a quantum counterpart to this approach, we use the extractable QROM formalism [DFMS22].

This models quantum-accessible random oracles like $\mathtt{H}$ as an extractable compressed random oracle $\mathtt{eCO}$ that has two interfaces, random oracle interface $\mathtt{eCO.RO}$ and an extraction interface $\mathtt{eCO.E}_f$ that is defined relative to a function $f : X \times Y = T$, where X and Y are the random oracle's domain and co-domain, respectively, and T is some other set. For our purposes, we identify X with the domain of $\mathtt{H}$, so $X := \mathcal{K} \times \mathcal{D} \times \mathcal{PK}^2 \times \mathcal{C}$, and $Y := T := \mathcal{T} \times \mathcal{SK}$, the tag and session key space. Since we want to isolate the first part of the output, we choose the function f defined by the projection onto the tag space $f(input, y = (tag, K)) = tag$. Extraction interface $\mathtt{eCO.E}_f$ takes as input a classical value $t \in T$. It does a quantum analogue to going through the random oracle queries and returning an x such that $f(x, \mathtt{H}(x)) = t$: it performs suitable measurements that collapse the oracle database, just enough so that the classical procedure would yield one particular outcome x for all parts of the superposition. For our choice of f, $\mathtt{eCO.E}_f(tag)$ simply returns a random oracle pre-image of $tag||sk$ for any sk.

Equipped with this extraction interface, the game waits until the attacker has finished and then calls $\mathtt{eCO.E}_f$ on all 'dishonest' tries, i.e., all tags *tag* that were received by an honest initiator whose password was not corrupted, without *tag* having been computed during a preceding honest execution of the respective responder. The game aborts if there exists a dishonest *tag* such that $\mathtt{eCO.E}_f(tag)$ returns a pre-image (K, pw, apk, pk', c) such that (K, apk, pk', c) matches the values which the initiator computed during the session to which the *tag* belongs. Since we already ruled out random oracle collisions, this in particular means that *pw* is the correct password. We denote this event by corrPW. We note that the extraction queries themselves do not change $\mathcal{A}$'s view since they are performed only after $\mathcal{A}$ finished [DFMS22, Th. 4.3, item 1].
We can now conclude that

$$\mathbf{Adv}_8 \leq \Pr[\,\mathsf{corrPW}\,]_{\mathbf{G}_8} \ .$$

Like in previous (classical) proofs, the change of probability for corrPW can be traced through the subsequent games $\mathbf{G}_8$-$\mathbf{G}_{15}$, by building anonymity and indistinguishability reductions that define their output based on corrPW.

Game $\boldsymbol{G}_9$: Notice correct pw guess already during game. In this game, we move the identification of corrPW into the game: instead of performing the extraction queries determining corrPW after running the attacker, we perform them already during the session runs. We perform them whenever the attacker sends a tag *tag* to an honest initiator whose password was not corrupted, without *tag* having been computed during a preceding honest execution of the respective responder. This change is in preparation for the following two game hops, where the reductions use corrPW to detect certain edge-cases of adversarial input that need to be addressed during the game.

To argue that this introduces little change, we use [DFMS22, Th. 4.3]: by properties 2.b and 2.c of Th. 4.3, any two subsequent queries to $\mathtt{eCO.RO}$ and $\mathtt{eCO.E}_f$ $8\sqrt{2\Gamma(f)/|\mathcal{K}| \cdot |\mathcal{T}|}$-almost-commute,

where

$$\Gamma(f) = \max_t \max_x |\{y \mid f(x,y) = t\}|, \tag{1}$$

which for our choice $f(x,(tag,K)) = tag$ equals $|\mathcal{K}|$. This means that we can commute the extraction queries into the game to right after when the respective Send query was received. We need to commute at most $n_a \cdot (n_s + q_{\mathtt{H}} H)$ many times until the extractions are at the right place, thus $\mathbf{Adv}_{\mathbf{9}} \leq n_a \cdot (n_s + q_{\mathtt{H}})\sqrt{2/|\mathcal{T}|}$ and also $|\Pr\left[\mathsf{corrPW}_{\mathbf{G}_8}\right] - \Pr\left[\mathsf{corrPW}_{\mathbf{G}_9}\right]| \leq n_a \cdot (n_s + q_{\mathtt{H}})\sqrt{2/|\mathcal{T}|}$.

Game $\boldsymbol{G}_{10}$: Randomize public-key pk. In this game, we randomize the public key used to create the responder ciphertext. First we recall the setting: in order to unlink the first (apk) and second (c, tag) protocol message, we argue that the ciphertext c does not leak which ciphertext was used to create it. More formally, we replace the public key used to derive c with a fresh uniformly random one, and reduce the distance in the games induced by this change to the ANO-PCA security of KEM.
We give a reduction that simulates the PAKE game for an adversary $\mathcal{A}$ that distinguishes BPR game $\mathbf{G}_{10}$ from $\mathbf{G}_8$ and uses $\mathcal{A}$ to solve its ANO-PCA challenge. Consider the different query patterns that the reduction has to address. First we note that sessions that are passively attacked (queries to `Execute`) or those where the initiator is honest can always be simulated without the use of programming in the ideal cipher: honest parties simply use the challenge public keys in place of their honest keys.
The more involved case is the one where the responder is honest, but the initiator is not: this setting involves an *adversarially chosen* encrypted public key apk that is decrypted by the honest responder into a public key. In order to move to game $\mathbf{G}_{10}$, where ciphertexts c are independent of the real public keys (or more accurately, the real password), the reduction reprograms decrypted public keys to challenge public keys in the ANO-PCA game.
We now show this proof step using only reprogramming according to the capabilities of SIM described above.

In a first step, the reduction answers all queries to E and D using the respective oracles of SIM. This change to SIM is not perfectly indistinguishable, therefore we have $\mathbf{Adv}_{10,1} \leq \varepsilon_{\mathsf{SIM}}$ where $\varepsilon_{\mathsf{SIM}}$ by definition of SIM.

The challenge set $\mathcal{X}$ used to initiate SIM is equal to the first set of public keys $\{pk_{0,0}, pk_{0,1}, \ldots pk_{0,n_a}\}$ provided by the anonymity challenger. This is the set that indicates the 'real' public keys, while the set $\{pk_{1,0}, pk_{1,1}, \ldots pk_{1,n_a}\}$ indicates the independent keys used *after* the reduction. Now, whenever the adversary submits a value apk for a session associated with password pw and key $k \leftarrow \mathtt{KDF}(pw)$, apk decrypts to a challenge public key whenever SIM programs successfully For $n_a$actively attacked sessions, we have by definition of SIM that

$$\Pr[(\vec{apk}, \vec{sid}) \leftarrow \mathcal{A}^{\mathsf{SIM}(\mathcal{X})} : \mathsf{D}_{\mathsf{SIM}}(k, apk_i) \in \mathcal{X} \forall (sid_i, apk_i) \in (\vec{sid}, \vec{apk_i})] = 1 - \delta_{\mathsf{SIM}}.$$

The reduction, on input apk, therefore decrypts apk to some challenge public key $pk_{0,j} \in \mathcal{X}$ and queries its challenger for a fresh anonymity challenge c related to that public key, and uses that in its response. That way, the adversary receives a ciphertext matching either pk_0 or pk_1 in each session, exactly as in the two games.

Finally, we note that since the reduction simulates all of $\mathcal{A}$'s oracles, so there is an edge case to consider: the case where $\mathcal{A}$ corrupts a party after it has queried the $\mathtt{Send}^0$ and $\mathtt{Send}^1$, but before it queries $\mathtt{Send}^2$ for matching sessions while forwarding messages. In this case, the oracle $\mathtt{Send}^2$ uses a challenge public key pk_0 that the attacker now *knows* due to the corruption. However, the reduction does not have the secret key associated with the public key, and it cannot decapsulate the ciphertext

and recompute the tag. Therefore, when the oracle decides whether or not to reject an adversarially generated (c, tag), the reduction has use the extraction interface of the extractable random oracle $\texttt{eCO}$ to determine the pre-key that was used by the adversary. Then, the reduction can query its plaintext-checking oracle to decide if the message should be rejected or not. Since we switched to the extractable oracle in a previous game, this does not incur any additional loss here.

Finally, the reduction outputs 0 whenever $\mathcal{A}$ wins, and 1 else. Therefore, the reduction wins whenever $\mathcal{A}$ wins and SIM successfully programmed for all its queries and we have $\mathbf{Adv}_{10,2} \leq \frac{1}{(1-\delta_{\mathsf{SIM}})} \cdot \mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}}$. We then arrive at the bounds

$$\mathbf{Adv}_{10} \leq \mathbf{Adv}_{10,1} + \mathbf{Adv}_{10,2} \leq \frac{1}{(1-\delta_{\mathsf{SIM}})} \cdot \mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}} + \varepsilon_{\mathsf{SIM}}.$$

$$|\Pr\Big[\ \mathsf{corrPW}\ _{\mathbf{G}_9}\Big] - \Pr\Big[\ \mathsf{corrPW}\ _{\mathbf{G}_{10}}\Big]| \leq \tfrac{1}{(1-\delta_{\mathsf{SIM}})} \cdot \mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}} + \varepsilon_{\mathsf{SIM}}.$$

Game $\boldsymbol{G}_{11}$: Randomize pre-key K. (Unchanged) For all queries to the $\texttt{Send}$ or $\texttt{Execute}$ oracles where flag $\texttt{trivGuess}$ is not raised before the query, we now randomize the pre-key K that is used to derive the final session key and the responder tag. For more details, see the pseudo-code in Figure 17. This change makes the pre-key independent of the ciphertext and the password for all fresh sessions. As previously, the change in success probability can be bounded using the indistinguishability advantage of KEM: $\mathbf{Adv}_{11} \leq n_s \cdot (n_a + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM}}$.

Game $\boldsymbol{G}_{12}$: Move dishonest tag identification back to end of game. In this game, we switch back to offline extraction, i.e., we only perform the extraction queries determining if a tag was valid after running the attacker. This is done in preparation for the next step that relies on measuring the compressed database at the end of the game. With the same reasoning as for game $\mathbf{G}_9$, we find

$$\mathbf{Adv}_{12} \leq n_a \cdot (n_s + q_{\texttt{H}})\sqrt{2/|\mathcal{T}|}$$

and

$$|\Pr\Big[\ \mathsf{corrPW}\ _{\mathbf{G}_{11}}\Big] - \Pr\Big[\ \mathsf{corrPW}\ _{\mathbf{G}_{12}}\Big]| \leq n_a \cdot (n_s + q_{\texttt{H}})\sqrt{2/|\mathcal{T}|}\ .$$

Game $\boldsymbol{G}_{13}$: Randomize Tag & Session Key. In the penultimate step of the proof, we randomize the responder tag tag and the session key SK, i.e., we simply sample random values from $\mathcal{T}$ and $\mathcal{SK}$, and return them as tag and session key. Recall, that in Game $\mathbf{G}_{12}$ the tag and session key are computed as $(tag, SK) \leftarrow \texttt{H}(K_{\$}; pw, apk, pk', c)$ where $K_{\$}$ is a fresh uniformly random sample from $\mathcal{K}$ that is not used anywhere else in the protocol. Hence, the values in both games follow the same marginal distribution. However, in Game $\mathbf{G}_{12}$ they are consistent with the random oracle, while in Game $\mathbf{G}_{13}$ they are not.

Intuitively, an adversary can only notice this inconsistency in Game $\mathbf{G}_{13}$ if it ever learned the value of $\texttt{H}$ at position $x^* = (K_{\$}; pw, apk, pk', c)$. Given that we simulate $\texttt{H}$ using the compressed oracle technique, this means that the difference between this and the last game can be bounded by the probability that the final database D of the compressed oracle in Game $\mathbf{G}_{13}$ contains a value for x^* when measured in the end, i.e.,

$$\mathbf{Adv}_{13} \leq \Pr[D(x^*) \neq \bot].$$

It remains to quantify $\Pr[D(x^*) \neq \bot]$. For this, note that in Game $\mathbf{G}_{13}$, $K_{\$}$ is not used anywhere in the protocol anymore and therefore the whole view of the adversary is independent of $K_{\$}$. For that

reason, we can delay the sampling of $K_{\$}$ until after we measured the database D. The database D has $q_{\mathtt{H}}$ entries. The probability that one of them is of the form $(K_{\$}; pw, apk, pk', c)$ for any value of $K_{\$}$ is therefore upper bounded by $1/|\mathcal{K}|$ and therefore

$$\mathbf{Adv}_{13} \leq \Pr[D(x^*) \neq \perp] \leq \frac{q_{\mathtt{H}}}{|\mathcal{K}|}.$$

Game $\boldsymbol{G}_{14}$: Randomize Session Key is folded into the previous game.

Game $\boldsymbol{G}_{15}$: Randomize Passwords. It remains to upper-bound corrPW. In this game, the attacker's view is completely independent of the chosen passwords, up to corrupted ones. This means that we can replace all non-corrupted passwords with fresh ones at the end, after running the attacker, and defining event corrPW with respect to the resampled ones. We note that there exists exactly one pre-image (K, pw, apk, pk', c) per tag that could trigger corrPW, and that the simulation SIM does not raise this flag as it did in the previous proof.

We can now bound the probability of corrPW in game $\mathbf{G}_8$, using the number of send queries and the password distribution. Assuming a uniform distribution on a password dictionary of size $|\mathcal{D}|$, and upper-bounding $\mathcal{A}$'s number of send queries by n_a,

$$\Pr[\,\mathsf{corrPW}_{\mathbf{G}_{15}}] \leq \frac{n_a}{|\mathcal{D}|}$$

And therefore $\Pr[\,\mathsf{corrPW}_{\mathbf{G}_8}] \leq \sum_{i=8}^{15} \Pr[\,\mathsf{corrPW}_{\mathbf{G}_i}]$.

Since the passwords and session keys are completely random from the adversary's view, we have that $|\Pr[\mathbf{G}_{15}(\mathcal{A}) \Rightarrow 1]| = \frac{1}{2}$. We can finally sum up the terms and state the bound:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{BPR}}_{\mathsf{OCAKE}}(\mathcal{A}) &\leq \sum_{i=0}^{15} |\Pr[\mathbf{G}_i(\mathcal{A}) \Rightarrow 1]| \\
&\leq \frac{n_a}{|\mathcal{D}|} + n_a^2 \cdot \eta_{\mathsf{KGen}} + \frac{|\mathcal{D}|^2}{|\mathcal{K}_{pw}|} + \frac{80e^2 q_{\mathtt{H}}^3 + 2}{|\mathcal{T}|} + n_a \cdot \delta^{\mathsf{KEM}}_{corr} + \\
&\quad + 4 \cdot n_a \cdot (n_s + q_{\mathtt{H}}) \sqrt{2/|\mathcal{T}|} + 2 \cdot \left(\frac{1}{(1 - \delta_{\mathsf{SIM}})} \cdot \mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}} + \varepsilon_{\mathsf{SIM}}\right) \\
&\quad + 2 \cdot n_s \cdot (n_a + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM}} + 2 \cdot \frac{q_{\mathtt{H}}}{|\mathcal{K}|}
\end{aligned}$$

4.2 Security of CAKE under SIM

Security of the CAKE protocol has been shown in the ROM and ICM in the universal composability framework [BCP+23] and in the BPR model [PZ23]. In the proof of [PZ23], there are four game hops that make use of programmability.[5] Additionally, in game $\mathbf{G}_7$ it is argued that reductions make arguments conditioned on values that were queried to the encryption oracles E_1 and E_2. Therefore, replacing the cipher with the simulator does not immediately result in a complete security proof. We therefore leave it as an open problem to construct such a proof
However, table 7 gives an overview of the game hops and sketches which games could potentially be instantiated with SIM.

[5] Appendix B gives a more thorough analysis of the necessary changes and the role of the ideal cipher in the proof.

Game	Description	Update?	Loss
$\mathbf{G}_{-1}$	Original BPR game		
$\mathbf{G}_0$	Collision Events.	No change.	$S^2(\eta_{pk} + \eta_{ct}) + \frac{(q_1^2+S^2)}{\lvert\mathcal{E}_1\rvert} + \frac{(q_2^2+S^2)}{\lvert\mathcal{E}_2\rvert} + \frac{q_1^2}{\lvert\mathcal{PK}\rvert} + \frac{q_2^2}{\lvert\mathcal{C}\rvert} + \frac{q_{\mathrm{H}}^2+S^2}{\lvert\mathcal{SK}\rvert}$
$\mathbf{G}_1$	Freshness.	No change.	0
$\mathbf{G}_2$	Sample D_1 using KGen.	Use SIM.	-
$\mathbf{G}_3$	Randomize Session keys (passive).	No change.	$\mathbf{Adv}^{\mathsf{(S,1)\text{-}OW\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{B}_2)$
$\mathbf{G}_4$	Randomize ciphertexts (passive).	No change.	$\mathbf{Adv}^{\mathsf{(S,1)\text{-}ANO}}_{\mathsf{KEM}}(\mathcal{B}_3)$
$\mathbf{G}_5$	Randomize protocol messages (passive).	No change.	0
$\mathbf{G}_6$	Do not decapsulate honest.	No change.	$S \cdot \delta$
$\mathbf{G}_7$	Extract password.	May require extraction.	open
$\mathbf{G}_8$	Randomize pre-key.	Use SIM.	$(1-\delta_{\mathsf{SIM}}) \cdot \mathsf{OW\text{-}rPCA} + \varepsilon_{\mathsf{SIM}}$.
$\mathbf{G}_9$	Randomize pre-key.	Use SIM.	$(1-\delta_{\mathsf{SIM}}) \cdot \mathsf{OW\text{-}PCA} + \varepsilon_{\mathsf{SIM}}$.
$\mathbf{G}_{10}$	Randomize public keys.	Use SIM.	$(1-\delta_{\mathsf{SIM}}) \cdot \mathsf{ANO\text{-}PCA} + \varepsilon_{\mathsf{SIM}}$.
$\mathbf{G}_{11}$	Randomize protocol messages (active).	No change.	0
$\mathbf{G}_{12}$	Randomize passwords	No change.	$\frac{S}{\lvert D\rvert}$

Fig. 7. Overview of proof steps in the proof of CAKE. Proof steps highlighted in grey require changes to consider quantum adversaries. These updated arguments are left as an open problem, but games $\mathbf{G}_2$, $\mathbf{G}_8$-$\mathbf{G}_{10}$ can likely be instantiated using SIM.

5 Necessity of SIM for OCAKE

We now study under which conditions any proof of security of the (O)CAKE protocol in the BPR model necessitates the use of programming. To that end, we first show that anonymity of the underlying KEM is a necessary property by describing an attack on the CAKE and OCAKE protocols that leverages non-anonymity. Secondly, we show using the meta-reduction technique how a successful reduction from the security of OCAKE to anonymity must program the ideal cipher oracle.

Anonymity is Necessary. In PAKE proofs, it is important to rule out dictionary attacks that would allow an attacker to use offline computations to determine the password used in some transcripts. This is why there is often a step that un-links the protocol messages from each other and the password. An example is the case of anonymity for OCAKE: the first message is an encryption of an ephemeral public key under the password. If an adversary were able to extract that public key from the second message, this would lead to a dictionary attack on the first message. To formalize this intuition, we show how to translate an attack on anonymity of KEM into an attack against the security of OCAKE. Let $\mathcal{B}$ be an algorithm in the anonymity game against KEM with advantage $\mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{B})$. Recall that by definition, whenever $\mathcal{B}$ receives some ciphertext c, $\mathcal{B}$ can determine which of two public keys pk_0, pk_1 was used to create that ciphertext with advantage

$$\mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{B}) = \lvert \Pr[\mathsf{ANO\text{-}PCA}^0(\mathcal{B}) \Rightarrow 0] - \Pr[\mathsf{ANO\text{-}PCA}^1(\mathcal{B}) \Rightarrow 0] \rvert$$

i.e., $\mathcal{B}$ outputs 0 whenever it guesses pk_0 was used, and 1 else.
We use the convention that if $\mathcal{B}$ is run with two public keys s.t. neither or both are associated with the challenge ciphertext, $\mathcal{B}$ outputs a random bit. We argue that $\mathcal{B}$ can be used to launch an attack on the protocol.
We describe the adversary $\mathcal{A}$ against the security of OCAKE. To launch the attack, $\mathcal{A}$ first obtains

some transcript apk, c, tag. $\mathcal{A}$ then chooses two passwords pw, pw' from the dictionary[6] and queries $\mathsf{D}(pw, apk)$ and $\mathsf{D}(pw', apk)$ to receive candidate public keys pk, pk'.
Next, $\mathcal{A}$ runs $\mathcal{B}$ on challenge values c, pk, pk' to receive a bit b. If $\mathcal{B}$'s output is 0, $\mathcal{A}$ selects pw as its guess pw^*, and pw' else.
Finally, $\mathcal{A}$ finishes the attack by running the sender side of the protocol using pw^* and sending the output to the receiver. The probability of $\mathcal{A}$ using the correct password is therefore increased by the anonymity advantage of $\mathcal{B}$ in all cases where one of the two passwords was the correct one for that session. This event represents a successful break of the protocol as the adversary using the correct password derives a shared key with a receiver or sender without them aborting. This attack can be improved by running the test on all pairs of passwords in the dictionary, ensuring that the real public key was queried, and taking a majority vote. This is shown in the algorithm in figure 8.
If $\mathcal{B}$ can distinguish a public key used to create c from a random public key with advantage at least ε, then there is one public key that is chosen $(|\mathcal{D}| - 1)(1 + \varepsilon)/2$ times in the comparisons, while all others are chosen only $(|\mathcal{D}| - 1)/2$ times in expectation. For noticeable ε this results in a noticeable advantage for $\mathcal{A}$.

Any successful anonymity reduction must program. We now use a meta-reduction technique to investigate how any successful reduction between the security of OCAKE and the anonymity of KEM must program the implementation of the block cipher (or its modeling). More precisely, we argue that there can be no reduction from the anonymity property of KEM to distinguishing the game with random public keys from the previous game *unless* the reduction programs the block cipher in some form.
We show that any successful reduction of ANO-PCA security to PAKE-security that does not implement a programming simulator can be used by a meta-reduction to solve anonymity challenges for a certain class of KEM that we call `Valid`-resistant Anonymous KEM. Intuitively, this class of KEM achieves anonymity even when the adversary has access to a validity oracle that given a public key pk and a ciphertext c decides if c is a valid ciphertext under pk (rejecting if c or pk are part of the anonymity challenge). Moreover, we show that KEM obtained via the full FO transform (including de-randomization in what is commonly referred to as the T-transform [HHK17]) are `Valid`-resistant Anonymous KEM. In consequence, we show that

1. there cannot exist a generic reduction that reduces anonymity to PAKE security of OCAKE without programming,
2. there cannot exist a reduction that reduces anonymity to PAKE security of OCAKE without programming for any KEM obtained via the FO-transform, including Kyber / ML-KEM.

We now formalize these results, starting with the definition of the class of KEMs. In this definition, we relate the advantage of two adversaries against ANO-PCA (c.f., Fig. 2) where one gets access to the additional validation oracle mentioned above. This oracle is described in Fig. 9. In the definition of this oracle, we assume an unbounded challenger that can brute-force secret keys to find a valid secret key for a given public key. We denote this brute-force algorithm as compute_secret_key. We will later show that for KEM obtained via the FO-transform, one can efficiently simulate this oracle via the random oracle used in the transform to de-randomize the initial scheme.

Definition 10 (`Valid`-resistant Anonymous KEM). *Let* KEM *be a key encapsulation mechanism with public-key space $\mathcal{PK}$ and key space $\mathcal{K}$.* KEM *is* `Valid`-resistant anonymous *if the* ANO-PCA*-advantage of an adversary $\mathcal{B}$ with access to an additional oracle* `Valid` *as in Fig. 9 is close to that of a regular anonymity adversary $\mathcal{A}$ in the* ANO-PCA *game.*

[6] Note that since we are arguing necessity, it would be sufficient to look at the simplest case with one session and only two elements in the dictionary.

Algorithm `Attack`

41 $P, P' \leftarrow \mathcal{P}^2$ // Choose parties to attack
42 $apk \leftarrow\$ \mathcal{PK}$ $e_1 \leftarrow\$ \mathcal{PK}$ // Select the honest pk implicitly
43 $(c^*, tag^*) \leftarrow \mathtt{Send}^1(P', apk^*)$ $e_2 \leftarrow \mathtt{Send}^1(P', e_1)$
44
45 for pw in $\mathcal{D}$ // Obtain candidate public keys
46 $pk[pw] \leftarrow \mathsf{BC}^{-1}(pw, apk)$
47 for $pw \in \mathcal{D}$ // Compute pairwise guess
48 $\mathtt{count\_wins}[pw] = 0$
49 for all $c \leftarrow \mathsf{D}_2(pw, e_2)$:
50 for $(pw, pw') \in \mathcal{D}^2$
51 $\mathtt{winner} \leftarrow \mathcal{B}(c, pk[pw], pk[pw'])$
52 if $\mathtt{winner} == 0$
53 $\mathtt{count\_wins}[pw] += 1$
54 else //$\mathtt{winner} = 1$
55 $\mathtt{count\_wins}[pw'] += 1$
56
57 $pw^* \leftarrow \perp$ // Compute best guess
58 $\mathtt{max\_count} \leftarrow 0$
59 for $pw \in \mathcal{D}$
60 if $\mathtt{count\_wins}[pw] >= \mathtt{max\_count}$
61 $\mathtt{max\_count} = \mathtt{count\_wins}[pw]$
62 $pw^* \leftarrow pw$
63
64 $(pk^*, sk^*) \leftarrow \mathsf{KGen}$ // Attack using candidate password
65 $apk^* \leftarrow \mathsf{BC}(pw^*, pk^*)$ $e_1 \leftarrow\$ \mathcal{PK}$
66 $(c^*, tag^*) \leftarrow \mathtt{Send}^1(P', apk^*)$ $e_2 \leftarrow \mathtt{Send}^1(P', e_1)$
67 $c^* \leftarrow \mathsf{D}_2(pw^*, e_2)$
68 $K^* \leftarrow \mathsf{Decap}(sk^*, c^*)$
69 $SK \leftarrow \mathtt{KDF}(\ldots, K^*)$
70 $SK^* \leftarrow \mathtt{Test}(P')$
71 if $SK == SK^*$ return 0
72 else return 1

Fig. 8. Attack Algorithm for adversary $\mathcal{A}$ using anonymity adversary $\mathcal{B}$ to attack security of OCAKE and CAKE. The attack uses a single transcript to compute the most likely password, then actively attacks a session with that password. Lines 7 to 21 are used to find the best possible guess out of all passwords using an anonymity adversary that works on pairs of public keys.

```
Oracle Valid (c,pk)
01 sk ← pk.compute_secret_key                          //unbounded algorithm
02 k ← Decap*(sk, c)
03 if k ≠⊥ return True
04 else return False
```

Fig. 9. Validity oracle `Valid` outputs True if and only if c decrypts successfully under some secret key associated with public key pk. Decap^* is an algorithm that runs decapsulation, but indicates decryption failures even for implicitly rejecting schemes. This can be instantiated in various ways, e.g. using a re-encryption check.

We now state our main theorem:

Theorem 2. *Let* KEM *be a* `Valid`*-resistant anonymous, strongly robust and* $\delta_{\mathsf{KEM}}-$*correct KEM. Then there exists an efficient algorithm* $\mathcal{M}$ *called the meta-reduction that can use any reduction* $\mathcal{R}$ *reducing* ANO-PCA *to BPR security of OCAKE without implementing a programming simulator* SIM *to break* ANO-PCA *of* KEM*.*

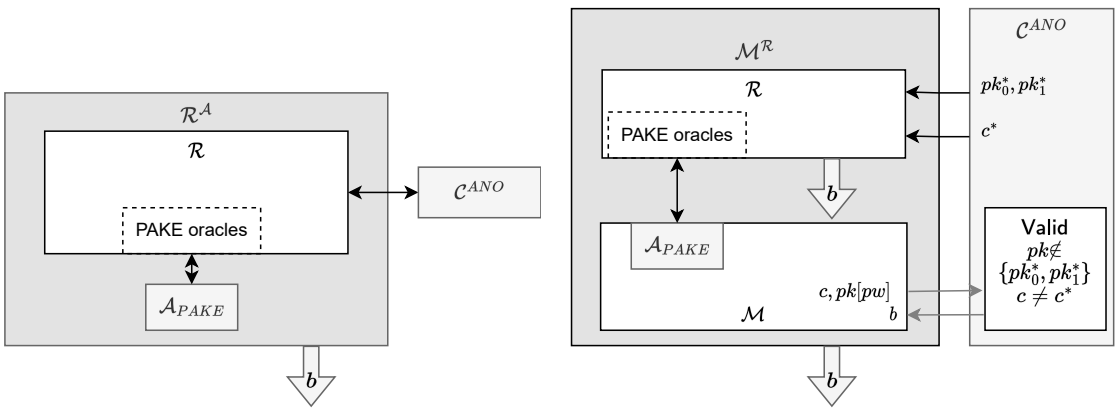
Theorem 2 essentially proves that any such reduction $\mathcal{R}$ has to be able to break anonymity itself. We construct the meta-reduction $\mathcal{M}$ that simulates a PAKE adversary $\mathcal{A}$ towards $\mathcal{R}$ as presented in pseudocode in Fig. 10. In our description we denote by $\mathcal{C}^{\mathsf{ANO}}$-PCA an anonymity challenger. We proceed in two steps. We first present $\mathcal{M}$ with respect to a $\mathcal{C}^{\mathsf{ANO}}$-PCA that provides an additional `Valid` oracle. Afterwards, we prove that for schemes that make use of the full FO-transform `Valid` can be simulated by $\mathcal{M}$ itself, turning $\mathcal{M}$ into a standard ANO-PCA adversary.

Proof outline. The correctness of the meta-reduction is shown in two parts: First we show that to the reduction, the meta-reduction behaves as an optimal anonymity adversary. For this we show how to implement the attack from above using the `Valid` oracle. Second, we show that the meta-reduction does not trivialize the challenge for the reduction by showing that the necessary oracles can be simulated when the KEM is obtained via the FO-transform, including the T-transform. From that it follows that there can be no general, non-programming reduction, not even a reduction that only applies to FO-transformed schemes.

The meta-reduction's adversary $\mathcal{A}$ is optimal. By definition, a PAKE adversary wins its game whenever it distinguishes a session key from random. We show the meta-reduction $\mathcal{M}$'s attack algorithm instantiated using the `Valid` oracle behaves as a successful PAKE adversary leveraging non-anonymity of the scheme. We summarize the stages of the attack. The attacker:

1. *generates* an authenticated public key apk,
2. *queries* the reduction's send oracle to receive (c, tag),
3. *determines* the most likely public key,
4. *derives* the password associated with that public key, and
5. *attacks* a session using that password.

To determine the most likely public key, the meta-reduction makes use of the `Valid` oracle, querying on the ciphertext c received from the reduction and each of the candidate public keys pk_{pw_i} associated with the different passwords.
First, we rule out a special case. Recall that the oracle `Valid` has a restriction: it *rejects* queries on challenge public keys. However, since $\mathcal{R}$ is non-programming according to the definition given by SIM, the public key associated with the password of the session cannot be equal to the challenge key in the anonymity question. Therefore, the success probability of the meta-reduction is not affected by this restriction.



Meta-reduction $\mathcal{M}^{\mathcal{R}}$

$\mathcal{R}$ .init()
$P, P' \leftarrow \mathcal{P}^2$ // Choose parties to attack
$apk \leftarrow\$ \mathcal{PK}$ // Select the honest pk implicitly
$(c^*, tag^*) \leftarrow \mathcal{R}.\mathtt{Send}^1(P', apk^*)$

for $pw \in \mathcal{D}$: if $\mathtt{Valid}(c, \mathsf{BC}^{-1}(pw, apk))$ set $pw^* \leftarrow pw$ // Compute guess

$(pk^*, sk^*) \leftarrow \mathsf{KGen}$ // Attack using candidate password
$apk^* \leftarrow \mathsf{BC}(pw^*, pk^*)$
$(c^*, tag^*) \leftarrow \mathcal{R}.\mathtt{Send}^1(P', apk^*)$
$K^* \leftarrow \mathsf{Decap}(sk^*, c^*)$
$SK \leftarrow \mathtt{KDF}(\ldots, K^*)$
$SK^* \leftarrow \mathcal{R}.\mathtt{Test}(P')$
if $SK == SK^*$ return 0 to $\mathcal{R}$, else return 1
output $b_{\mathcal{R}} \leftarrow \mathcal{R}$ // Output reduction's bit

Fig. 10. Left: Algorithm $\mathcal{R}^{\mathcal{A}}$ consisting of reduction $\mathcal{R}$ leveraging PAKE adversary $\mathcal{A}_{PAKE}$ to solve $\mathcal{C}^{\mathsf{ANO}}$'s challenges. Right: Algorithm $\mathcal{M}^{\mathcal{R}}$ consisting of meta-reduction $\mathcal{M}$ leveraging reduction $\mathcal{R}$ to solve $\mathcal{C}_{\mathsf{ANO}}$'s challenges. Bottom: Pseudocode for the meta-reduction algorithm using reduction $\mathcal{R}$ to break challenges.

To restate: we eliminate from the analysis the case where the decryption of apk under the correct password matches the challenge public key pk_0, since it implies that $\mathcal{R}$ is programming.

`Valid` oracle solves anonymity if KEM is SROB and correct Secondly, we show that the `Valid` oracle solves anonymity whenever KEM is strongly robust (SROB) and correct. [7]
Giving a single example of a scheme s.t. the meta-reduction holds shows that that there can be no *generic* non-programming reduction. Additionally, we argue that the class of schemes for which the meta-reduction holds includes many that are of practical relevance in post-quantum applications. A particularly relevant example is Kyber, recently standardized as ML-KEM, which is both anonymous and strongly robust ([GMP22],[MX23]).
To show that the `Valid` oracle solves anonymity, we argue that the `Valid` oracle outputs `True` if and only if it is queried on a matching ciphertext-public key pair.

[7] The alternative is the case where anonymity holds even against information-theoretic attackers, and the ciphertext distribution is identical across key pairs.

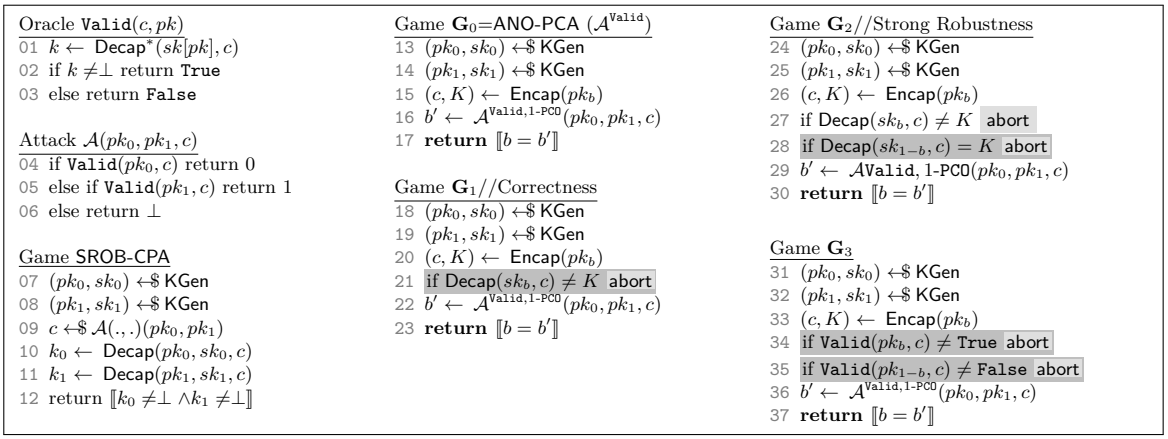


Fig. 11. Game hops in the proof of optimality of an attacker utilizing the Valid oracle to attack anonymity. Game $\mathbf{G}_0$ matches the anonymity game, while the final game $\mathbf{G}_3$ matches the attack algorithm.

Lemma 2 (Valid oracle solves anonymity if KEM is SROB and correct). *For* KEM *that is strongly robust and* $\delta_{\mathsf{KEM}}-$*correct, an attacker with access to a* Valid *oracle (Fig. 9) can win the anonymity game. More formally, we show that* $\mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{A}^{\mathtt{Valid}}) = 1 - \mathbf{Adv}^{\mathsf{SROB}}_{\mathsf{KEM}}(\mathcal{B}) - \delta_{\mathsf{KEM}}$.

Proof (Lemma 2). We prove Lemma 2 using a game-hopping proof in four games. Starting from the ANO-PCA game, with the additional oracle, we modify the game by first ruling out correctness errors. For KEM that is $\delta^{\mathsf{KEM}}_{corr}-$correct, we bound the distance to the previous game as $|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq \delta^{\mathsf{KEM}}_{corr}$. Secondly we reduce the distance between games $\mathbf{G}_1$ and $\mathbf{G}_2$ to the strong robustness property of KEM: $|\Pr[\mathbf{G}_1 \Rightarrow 1] - \Pr[\mathbf{G}_0 \Rightarrow 1]| \leq \mathbf{Adv}^{\mathsf{KEM}}_{\mathsf{SROB}}$. Finally, we switch the decapsulation computations to calls to the Valid oracle. Since the computations in the game then match the attack, we see that the attacker in game $\mathbf{G}_3$ always wins: $\Pr[\mathbf{G}_3 \Rightarrow 1] = 1$. Figure 11 shows pseudocode for all game hops in the proof.

$$\Pr[\mathbf{G}_0] = \underbrace{\Pr[\mathbf{G}_3]}_{1} - \underbrace{(\Pr[\mathbf{G}_3] - \Pr[\mathbf{G}_2])}_{0} - \underbrace{(\Pr[\mathbf{G}_2] - \Pr[\mathbf{G}_1])}_{\mathbf{Adv}^{\mathsf{SROB}}_{\mathsf{KEM}}(\mathcal{B})} - \underbrace{(\Pr[\mathbf{G}_1] - \Pr[\mathbf{G}_0])}_{\delta}$$

$$\mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{A}^{\mathtt{Valid}}) = 1 - \mathbf{Adv}^{\mathsf{SROB}}_{\mathsf{KEM}}(\mathcal{B}) - \delta$$

Anonymity is preserved under Validity oracle. To show that the validity oracle does not solve the challenge for the reduction, we investigate when ANO-PCA security is preserved under addition of the validity oracle. We formalize this property as validity resistance. Finally, we show that for certain schemes any successful adversary that makes use of the validity oracle can be converted into a general anonymity adversary. Finally, we give a reduction in the quantum random oracle model for the case of FO transformed schemes.

Lemma 3 (FO-transformed schemes are Valid-resistant Anonymous.). *Let* KEM *be constructed using the T-transform [HHK17] from a PKE that is* $\gamma-$*spread and* $\delta^{\mathsf{PKE}}_{corr}-$*correct.*
Then access to an oracle Valid *(Fig. 9) does not significantly improve the advantage of an adversary in the* ANO-PCA *game.*
Concretely, for an ANO-PCA *adversary* $\mathcal{A}^{\mathtt{Valid}}$ *that additionally has access to the validity oracle* Valid*, making* q_V *many queries to* Valid *,* q_P *many queries to the plaintext checking oracle* 1-PCO *and* q_G *queries to random oracle* G *(the one used to derive randomness), we can construct an*

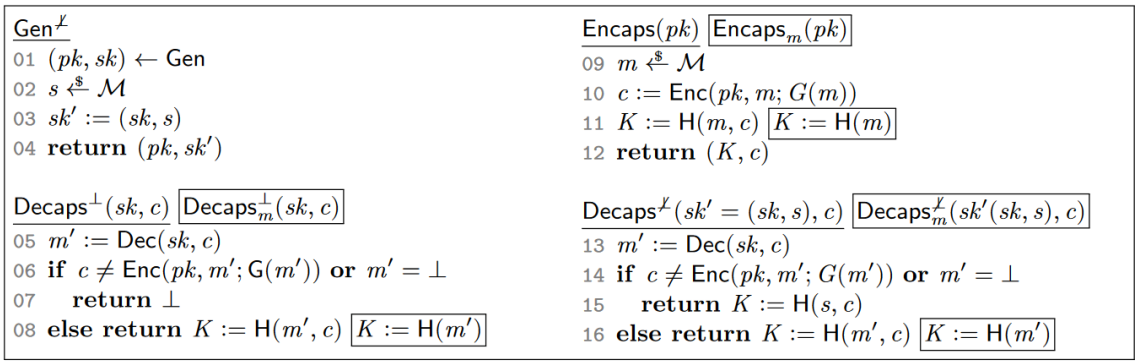


Fig. 12. Algorithms of FO transformed KEM according to [HHK17].

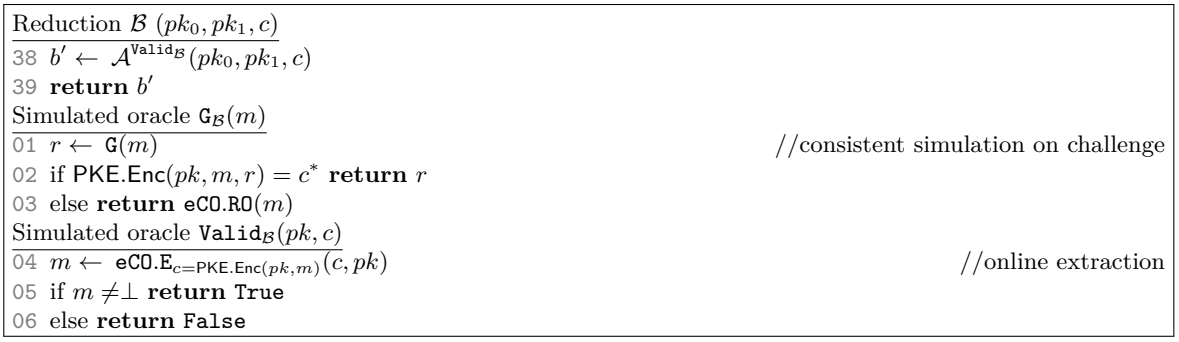


Fig. 13. The simulated oracles of Adversary $\mathcal{B}$ reducing security under the additional validity oracle to ANO-PCA security.

ANO-PCA *adversary* $\mathcal{B}$ *without access to* `Valid` *s.t.*

$$\mathbf{Adv}^{\mathsf{KEM}}_{\mathsf{ANO\text{-}PCA}}(\mathcal{A}^{\mathtt{Valid}}) \leq \mathbf{Adv}^{\mathsf{KEM}}_{\mathsf{ANO\text{-}PCA}}(\mathcal{B}) + q_V \cdot \delta^{\mathsf{PKE}}_{corr} + q_V \cdot 2^{-\gamma} + 12 q_V (q_V + q_P + q_G) 2^{-\frac{\gamma}{2}}.$$

Proof (Lemma 3). We show that ANO-PCA security implies security under the addition of the `Valid` oracle, using a reduction in the QROM. Adversary $\mathcal{B}$ runs $\mathcal{A}^{\mathtt{Valid}}$ and simulates its oracles by forwarding the challenge public keys and the queries to the challenge and plaintext-checking oracles to its ANO-PCA challenger. When $\mathcal{A}^{\mathtt{Valid}}$ outputs a bit, $\mathcal{B}$ forwards it to the ANO-PCA challenger and wins whenever $\mathcal{A}^{\mathtt{Valid}}$ does.
Adversary $\mathcal{B}$ simulates the validity oracle $\mathtt{Valid}_{\mathcal{B}}$ and random oracle $\mathtt{G}_{\mathcal{B}}$ as shown in Fig. 13. First we note that there are two cases where the simulated oracle behaves differently from the way is defined for an unbounded adversary:

- Case 1: $\mathtt{Valid}_{\mathcal{B}}$ returns `False` while `Valid` does not. This case occurs whenever the adversary has found m s.t. $c = \mathsf{PKE.Enc}(pk, m, \mathtt{G}(m))$ without querying `G` on m. Using Lemma 1 of [HHM22] and noticing that the simulated `Valid` oracle implements a decapsulation oracle with reject internally, we can bound the probability of this event as $q_V \cdot 2^{-\gamma}$ when PKE is $\gamma-$spread.
- Case 2: $\mathtt{Valid}_{\mathcal{B}}$ returns `True` while `Valid` does not. This case corresponds to a correctness error of the KEM: $\mathcal{A}^{\mathtt{Valid}}$ has found a ciphertext s.t. $\mathsf{Decap}(sk, c) \neq K$ for $(c, K) \leftarrow\!\$\, \mathsf{Encap}(pk)$. We can bound the probability of this event $q_V \cdot \delta^{\mathsf{PKE}}_{corr}$.

Secondly, the simulated oracle extracts messages from the random oracle that match the ciphertext. To argue about this we again make use of the extractable QROM formalism [DFMS22] as well Lemma 1 of [HHM22].
The simulated $\mathtt{Valid}_{\mathcal{B}}$ oracle uses the extraction interface $\mathtt{eCO.E}_f$ defined relative to a function $f : X \times Y = T$, where X and Y are the random oracle's domain and co-domain, respectively, and T is some other set. Here, we use the encryption function $\mathsf{PKE.Enc}$ (pk,m,r) as our function f, i.e., we extract messages m s.t. for queried public key pk and ciphertext c, we have that $c = \mathsf{PKE.Enc}(pk, m, \mathtt{G}(m))$.
If the adversary has previously made a query that fulfills this predicate, the extraction returns this message m and the $\mathtt{Valid}_{\mathcal{B}}$ oracle returns true. In the case that no such query was made, the extraction returns $\perp$ and the $\mathtt{Valid}_{\mathcal{B}}$ oracle returns false.
Finally, we see using the technique from [HHM22] that the disturbance the online extraction incurs can be bounded as $8\sqrt{2\Gamma(f)/|\mathcal{R}|}$ with

$$\Gamma(f) = \max_t \max_x |\{y \mid R(x,y) = t\}| = 2^{-\gamma} \cdot |\mathcal{R}| \tag{2}$$

since $\mathtt{eCO.RO}$ and $\mathtt{eCO.E}_f$ almost-commute, and PKE is $\gamma-$spread.
We swap the $\mathtt{eCO.RO}$ call that produces m with all calls to $\mathtt{eCO.E}$ that happen after the adversary submits c, including the calls inside the validity oracle and the plaintext-checking oracle. Therefore, we can bound the distance between the games:

$$\mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{A}^{\mathtt{Valid}}) \leq \mathbf{Adv}^{\mathsf{KEM}}_{\mathsf{ANO\text{-}PCA}}(\mathcal{B}) + q_V \cdot \delta^{\mathsf{PKE}}_{corr} + q_V \cdot 2^{-\gamma} + 12 q_V (q_V + q_P + q_G) \cdot 2^{-\frac{\gamma}{2}}.$$

References

AHH+23. Nouri Alnahawi, Kathrin Hövelmanns, Andreas Hülsing, Silvia Ritsch, and Alexander Wiesmaier. Towards post-quantum secure pake-a tight security proof for ocake in the bpr model. *Cryptology ePrint Archive*, 2023.

AHHR24. Nouri Alnahawi, Kathrin Hövelmanns, Andreas Hülsing, and Silvia Ritsch. Towards post-quantum secure PAKE - A tight security proof for OCAKE in the BPR model. In Markulf Kohlweiss, Roberto Di Pietro, and Alastair R. Beresford, editors, *CANS 2014: 23rd International Conference on Cryptology and Network Security, Part II*, volume 14906 of *Lecture Notes in Computer Science*, pages 191–212, Cambridge, UK, September 24–27, 2024. Springer, Singapore, Singapore.

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: 21st International Conference on Applied Cryptography and Network Security, Part II*, volume 13906 of *Lecture Notes in Computer Science*, pages 516–538, Kyoto, Japan, June 19–22, 2023. Springer, Cham, Switzerland.

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, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 41–69, Seoul, South Korea, December 4–8, 2011. Springer Berlin Heidelberg, Germany.

Bla06. John Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In Matthew J. B. Robshaw, editor, *Fast Software Encryption – FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 328–340, Graz, Austria, March 15–17, 2006. Springer Berlin Heidelberg, Germany.

BPR00. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155, Bruges, Belgium, May 14–18, 2000. Springer Berlin Heidelberg, Germany.

BV98. Dan Boneh and Ramarathnam Venkatesan. Breaking RSA may not be equivalent to factoring. In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 59–71, Espoo, Finland, May 31 – June 4, 1998. Springer Berlin Heidelberg, Germany.

CFHL21. Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. On the compressed-oracle technique, and post-quantum security of proofs of sequential work. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 598–629, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland.

DAL+17. Jintai Ding, Saed Alsayigh, Jean Lancrenon, Saraswathy Rv, and Michael Snook. Provably secure password authenticated key exchange based on rlwe for the post-quantum world. In *Cryptographers' Track at the RSA conference*, pages 183–204. Springer, 2017.

DFMS22. Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Online-extractability in the quantum random-oracle model. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 677–706, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

DKBY24. Vivek Dabra, Saru Kumari, Anju Bala, and Sonam Yadav. Sl3pake: Simple lattice-based three-party password authenticated key exchange for post-quantum world. *Journal of Information Security and Applications*, 84:103826, 2024.

FF13. Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of Schnorr signatures. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 444–460, Athens, Greece, May 26–30, 2013. Springer Berlin Heidelberg, Germany.

FLR+10. Marc Fischlin, Anja Lehmann, Thomas Ristenpart, Thomas Shrimpton, Martijn Stam, and Stefano Tessaro. Random oracles with(out) programmability. In Masayuki Abe, editor, *Advances*

in Cryptology – ASIACRYPT 2010, volume 6477 of *Lecture Notes in Computer Science*, pages 303–320, Singapore, December 5–9, 2010. Springer Berlin Heidelberg, Germany.

GMP22. Paul Grubbs, Varun Maram, and Kenneth G. Paterson. Anonymous, robust post-quantum public key encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 402–432, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

HHK17. Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the Fujisaki-Okamoto transformation. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.

HHM22. Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz. Failing gracefully: Decryption failures and the Fujisaki-Okamoto transform. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022, Part IV*, volume 13794 of *Lecture Notes in Computer Science*, pages 414–443, Taipei, Taiwan, December 5–9, 2022. Springer, Cham, Switzerland.

HM24. Kathrin Hövelmanns and Christian Majenz. A note on failing gracefully: Completing the picture for explicitly rejecting fujisaki-okamoto transforms using worst-case correctness. In Markku-Juhani Saarinen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II*, pages 245–265, Oxford, UK, June 12–14, 2024. Springer, Cham, Switzerland.

LLH24. You Lyu, Shengli Liu, and Shuai Han. Universal composable password authenticated key exchange for the post-quantum world. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 120–150. Springer, 2024.

MX23. Varun Maram and Keita Xagawa. Post-quantum anonymity of Kyber. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023: 26th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 13940 of *Lecture Notes in Computer Science*, pages 3–35, Atlanta, GA, USA, May 7–10, 2023. Springer, Cham, Switzerland.

PZ23. Jiaxin Pan and Runzhi Zeng. A generic construction of tightly secure password-based authenticated key exchange. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part VIII*, volume 14445 of *Lecture Notes in Computer Science*, pages 143–175, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore.

SA23. Kübra Seyhan and Sedat Akleylek. A new password-authenticated module learning with rounding-based key exchange protocol: Saber. pake. *The Journal of Supercomputing*, 79(16):17859–17896, 2023.

SH19. Vladimir Soukharev and Basil Hess. Pqdh: a quantum-safe replacement for diffie-hellman based on sidh. *Cryptology ePrint Archive*, 2019.

TY19. Shintaro Terada and Kazuki Yoneyama. Password-based authenticated key exchange from standard isogeny assumptions. In *Provable Security: 13th International Conference, ProvSec 2019, Cairns, QLD, Australia, October 1–4, 2019, Proceedings 13*, pages 41–56. Springer, 2019.

Unr23. Dominique Unruh. Towards compressed permutation oracles. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part IV*, volume 14441 of *Lecture Notes in Computer Science*, pages 369–400, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore.

Xag22. Keita Xagawa. Anonymity of NIST PQC round 3 KEMs. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part III*, volume 13277 of *Lecture Notes in Computer Science*, pages 551–581, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland.

Zha19. Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 239–268, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

A (O)CAKE security and BPR model

In this section, we include the security model, as well as a complete proof, restating previous work in sections that are indicated by a frame.

A.1 Classical BPR proof for OCAKE

In this section, we restate the model and preliminaries given in [AHH+23]. Our security analysis is based on the BPR model for authenticated key exchange [BPR00]: security of a protocol Π is modeled through a security experiment in which the attacker interacts with oracles that represent honest parties (`Execute` and `Send`) as well as oracles that represent leakage of secret material (`Reveal` and `Corrupt`), and wins if it can distinguish an established session key from random. The involved oracles are described in more detail in Fig. 14. To exclude trivial attacks from consideration, [BPR00]

Query	Return Value	Description
$\texttt{Execute}(P, i, P', j)$	(apk, c, tag_1, tag_2)	Passive attack: Return transcript of an honest protocol execution between parties P and P', using the ith/jth session of P/P'.
$\texttt{Send}(P, i, msg, flow)$	msg'	Active attack: Send message msg to the oracle representing honest party P, causing it to proceed depending on its state. Flow indicator enumerates the messages in a run of the protocol and improves readability of the oracle.
$\texttt{Reveal}(P, i)$	$\mathrm{SK}[P, i]/\perp$	Session key leakage: Return session key SK of (P, i) iff (P, i) terminated, else $\perp$; marks this instance and its matching instance "unfresh".
$\texttt{Corrupt}(P, \texttt{PWD}')$	$\mathrm{PWD}[P, :]$	Password leakage or overwrite: Either return dictionary of passwords $\mathrm{PWD}[P, :]$ held by party P, or allow adversary to overwrite password dictionary with PWD'$[P, :]$.
$\texttt{Test}^b(P, i)$	$\mathrm{SK}[P, i]/SK^{\$}$	Session key challenge: Attack ith session of party P. Only for fresh, accepting instances. Returns either real or random session key depending on challenge bit b.
$\texttt{KDF}(pw)$	k_{pw}	Random oracle, input password $pw \in \mathrm{PWD}$, output Ideal Cipher key k_{pw}.
$\texttt{H}(msg)$	tag	Random oracle, input message msg, output $tag \in \mathcal{T}$.
$\texttt{KDF}'(msg)$	SK	Random oracle, input message msg, output session key $SK \in \mathcal{SK}$.
$\mathsf{E}(k, m)$	c	Ideal cipher encryption on input (key, message).
$\mathsf{D}(k, c)$	m	Ideal cipher decryption on input (key, ciphertext).

Fig. 14. Overview of the PAKE adversary's oracles provided by the security game. Top part (above double midrule): oracles present in the BPR model. Bottom part: Random oracles and ideal cipher oracles to which the attacker additionally has access to when attacking the OCAKE protocol.

define a freshness condition (Definition 12 below) that permits revealing a key on one side, and then testing the other (partnered) side, where partnered is defined as follows:

Definition 11 (Partnering). *Two instances $(P, i), (P', j)$ are partnered iff both instances have accepted (i.e. reached an `accept` instruction) with the same transcript and session key.*

Intuitively, 'unfreshness' expresses that the adversary may have learned the to-be-tested session's key SK in a trivial way, i.e., by having interacted with the oracles revealing secret information in a way such that SK becomes trivially derivable regardless of the protocol's nature. Concretely, the cases we cover in our freshness definition below are a), simply requesting the key from the `Reveal` oracle, and b), learning a password pw via `Corrupt` and then actively interfering with the test session, e.g., using pw to manipulate the peer into using a session key of the adversary's choosing.

Definition 12 (Freshness with Forward Secrecy). *Suppose that the adversary made exactly one `Test` query, and it was to party P and instance i. We say session i of party P is **unfresh** if at any time, there was a `Reveal` query to instance (P, i) above or the instance (P', j) that it is partnered with. We also say the session is **unfresh** if both the following conditions hold:*

- *Before the `Test` query, there was a `Corrupt` query on the test session's holder P or its partnered peer P'.*
- *One of the messages sent to P concerning the test session was* manipulated *by the adversary, i.e., there was a `Send`(P, i) query.*

*The session of (P, i) is only considered **fresh** if neither of these conditions are met.*

Experiment $\mathrm{Exp}^{\mathsf{BPR}}_{\Pi}(\mathcal{A})$

07 $b \xleftarrow{unif} \$ \{0,1\}$
08 $b' \leftarrow \mathcal{A}^{\mathcal{O}^b}(\mathcal{P})$
09 **return** $[\![b = b']\!]$

Fig. 15. The BPR security game for active adversaries. $\mathcal{O}^b$ = indicates the collection of oracles {`Execute`, `KDF`, `H`, `KDF'`, `E`, `D`, `Send`, `Reveal`, `Corrupt`,`Test`^{b}}. Here, $\mathcal{P}$ is the party set.

Definition 13 (Key indistinguishability of PAKE). *Let Π be a PAKE protocol. We say that an adversary $\mathcal{A}$, run in experiment Exp^{BPR}_{Π},* wins *if it correctly guesses the bit according to which the test query was defined and if the `Test` query was issued for a party (P, i) that has terminated and is fresh (see Definition 12). We define the* advantage of $\mathcal{A}$ against a PAKE protocol Π *as*

$$\mathbf{Adv}^{\mathsf{BPR}}_{\Pi}(\mathcal{A}) := |\Pr[\mathrm{Exp}^{\mathsf{BPR}}_{\Pi}(\mathcal{A}) \Rightarrow 1] - 1/2| \ .$$

Our modification of OCAKE uses key confirmation tags in both directions. While only the responder tag actually is needed for our security proof, we additionally include an initiator tag – following the 'add client-to-server authentication' (AddCSA) paradigm [BPR00] – to achieve explicit mutual authentication.

Definition 14 (Explicit Mutual Authentication). *A protocol achieves explicit mutual authentication if parties* accept *if and only if there exists a partnered party that accepts with the same output.*

A.2 Updated BPR proof for OCAKE

In this section, we fully flesh out our updated BPR security proof. Since we update the classical BPR proof, we work along the structure of that proof. For the reader's convenience, we thus repeat Fig. 6 from the main body that summarizes the classical proof's game-hops, which game-hops need updating, new proof techniques we use to do so, and the resulting loss.

Game	Description	Update?	Loss
$\mathbf{G}_0$	Original BPR game		
$\mathbf{G}_1$	Abort on KGen Collision.	Not necessary	$n_a^2 \cdot \eta_{\mathsf{KGen}}$
$\mathbf{G}_2$	Abort on KDF Collision.	Not necessary	$\frac{\lvert\mathcal{D}\rvert^2}{\lvert\mathcal{K}_{pw}\rvert}$
$\mathbf{G}_3$	IC lazy sampling w/ abort	This work (SIM)	-
$\mathbf{G}_4$	Prevent IC collisions	This work (SIM)	-
$\mathbf{G}_5$	Abort on Tag Collision.	This work (CO)	$\frac{80e^2 q_{\mathtt{H}}^3+2}{\lvert\mathcal{T}\rvert}$
$\mathbf{G}_7$	Do not decrypt honest c.	Not necessary	$n_a \cdot \delta_{corr}^{\mathsf{KEM}}$
$\mathbf{G}_6$	Sample IC using KGen	This work (SIM)	-
$\mathbf{G}_8$	Abort on tag under correct pw.	This work (eCO)	$\Pr[\,\mathsf{corrPW}\,]$
$\mathbf{G}_9$	Make eCO extractions online.	This work (eCO)	$n_a \cdot (n_s + q_{\mathtt{H}})\sqrt{2/\lvert\mathcal{T}\rvert}$
$\mathbf{G}_{10}$	Randomize public key.	This work (SIM)	$\frac{1}{(1-\delta_{\mathsf{SIM}})} \cdot \mathbf{Adv}_{\mathsf{KEM}}^{\text{ANO-PCA}} + \varepsilon_{\mathsf{SIM}}$
$\mathbf{G}_{11}$	Randomize pre-key.	Not necessary	$n_s \cdot (n_a + 1) \cdot \mathbf{Adv}_{\mathsf{KEM}}^{\text{IND-CPA}}$
$\mathbf{G}_{12}$	Move eCO extractions to end.	This work (eCO)	$n_a \cdot (n_s + q_{\mathtt{H}})\sqrt{2/\lvert\mathcal{T}\rvert}$
$\mathbf{G}_{13}$	Randomize Tag, Session Key.	This work (CO)	$\frac{q_{\mathtt{H}}}{\lvert\mathcal{K}\rvert}$
$\mathbf{G}_{15}$	Randomize Passwords.	Not necessary	0

In the following game hops, we will use the notational convention

$$\mathbf{Adv}_{\mathsf{i}} := \lvert\Pr[\mathbf{G}_i(\mathcal{A}) \Rightarrow 1]\rvert \ .$$

We indicate game-hops that need no update (and thus are copy-pasted) by 'restated'.

Game $\boldsymbol{G}_0$: Original BPR game. PAKE Oracles are exactly as in existing proof (but included in the appendix for reference), however H, KDF,KDF' and IC *are quantumly accessible.*

Game $\boldsymbol{G}_1$: Collision Handling KEM key generation. As previously, we have that $\mathbf{Adv}_1 \leq n_a^2 \cdot \eta_{\mathsf{KGen}}$.

BEGIN RESTATED

In this game, we abort whenever there are at least two sessions where the same ephemeral key pair (pk, sk) is sampled by the KEM key generation. Let η_{KGen}be the collision probability of KGen. Since games $\mathbf{G}_0$ and $\mathbf{G}_1$ are identical unless a collision occurs, we have that:

$$\lvert\mathbf{Adv}_0 - \mathbf{Adv}_1\rvert = \Pr[\mathtt{KDFColl}] \leq n_a^2 \cdot \eta_{\mathsf{KGen}}$$

END RESTATED

Game $\boldsymbol{G}_2$: Abort on KDF Collision. In this game, we abort whenever the key derivation function that maps passwords to block cipher keys outputs a collision on the password space.Since this bound is only dependent on the choice of KDF and dictionary $\mathcal{D}$, without any adversarial input, we do not have to model KDF as a random oracle and the bound is statistical. More precisely, if KDF is a function $f : \mathcal{D} \to \mathcal{K}_{pw}$ selected uniformly at random, the probability of it mapping any two inputs pw, pw' to the same output k_{pw} can be upper-bounded using a simple birthday collision bound: $\mathbf{Adv}_2 \leq \Pr[\,\mathsf{KDFColl}\,] \leq \frac{\lvert\mathcal{D}\rvert^2}{\lvert\mathcal{K}_{pw}\rvert}$. labelgame:appendix.IC1

Games $\boldsymbol{G}_3$ and $\boldsymbol{G}_4$ are folded into game $\mathbf{G}_{10}$. labelgame:appendix.IC2

Game $\boldsymbol{G}_5$: Abort on Tag Collision. One of the informal security goals of PAKE proofs is to rule out attacks where an adversary can *test* more than one password in a single online attack. One example of such an attack is the exploitation of a collision: if there should exist two passwords pw, pw' s.t. the responder tag *tag* computed from them for a given transcript is the same. Then the attacker could win the game by submitting *tag* without having uniquely identified the right password.
Therefore, in game $\mathbf{G}_5$, we abort whenever the attacker has found such a collision in $\mathtt{H}$. To argue about the distance between $\mathbf{G}_5$ and $\mathbf{G}_4$, we use the compressed oracle technique [CFHL21].
Let $\mathcal{A}$ be the adversary in the PAKE game. We define an adversary $\mathcal{B}$ in the collision-finding game that runs $\mathcal{A}$ and is successful if it outputs x, x' s.t. $\mathtt{H}(x) = \mathtt{H}(x')$ at the end of the game.
$\mathcal{B}$ simulates $\mathcal{A}$'s PAKE and random oracle queries. First, $\mathcal{B}$ replaces the random oracle with a compressed random oracle. The compressed random oracle allows $\mathcal{B}$ to record $\mathcal{A}$'s queries in a database.
Then, at the end of the game, $\mathcal{B}$ measures the database for collisions. Let R be the relation that describes a collision in the tag: for distinct x, x' and arbitrary sk, sk' we have that $\mathtt{H}(x)||sk = \mathtt{H}(x')||sk'$. Then we have $R \subseteq \mathcal{X}^l \times (\mathcal{T} \times \mathcal{SK})^l$ for $l = 2$.

$$\mathsf{CL} := \{D | \exists x \neq x' : D(x) = D(x') \neq \perp\}$$

Then, by Corollary 4.2 [CFHL21, Corollary 4.2], we have that for success probability in the original game p and success probability in the game using the compressed random oracle p', we have that $\sqrt{p} \leq \sqrt{p'} + \sqrt{\frac{2}{|\mathcal{T}|}}$.
We now argue about the success probability p', closely following the reasoning in section 2.3 of [CFHL21], (*Finding a Collision (with Parallel Queries)*). Let $\mathsf{SZ}_{\leq(s-1)}$ be a database of size at most $s-1$. We set $k = 1$ as we do not model parallel queries, so we have using the transition probability of a database that does not contain a collision to one that *does*

$$\sqrt{p'} \leq \sum_{s=1}^{q} [\![\mathsf{SZ}_{\leq(s-1)} \backslash \mathsf{CL} \to \mathsf{CL}]\!] \leq \sum_{s=1}^{q} 2e\sqrt{10\frac{q_{\mathtt{H}}}{|\mathcal{T}|}} \leq 2qe\sqrt{10\frac{q_{\mathtt{H}}}{|\mathcal{T}|}}$$

where e is Euler's number. Therefore we conclude $p' \leq \frac{40q^3e^2}{|\mathcal{T}|}$. From $\sqrt{p} \leq \sqrt{p'} + \sqrt{\frac{2}{|\mathcal{T}|}}$ we then see by squaring that

$$p \leq p' + \frac{2}{|\mathcal{T}|} + \sqrt{\frac{2p'}{|\mathcal{T}|}} \leq \frac{40e^2(q^3 + \sqrt{2}q^{3/2}) + 2}{|\mathcal{T}|} \leq \frac{80e^2q^3 + 2}{|\mathcal{T}|}$$

Finally, this allows us to bound the distance:

$$\mathbf{Adv}_5 \leq p \leq \frac{80e^2q_{\mathtt{H}}^3 + 2}{|\mathcal{T}|}.$$

labelgame:appendix.IC3

Game $\boldsymbol{G}_6$ is also folded into game $\mathbf{G}_{10}$.

Game $\boldsymbol{G}_7$: Correctness. As previously, $\mathbf{Adv}_7 \leq n_a \cdot \delta_{corr}^{\mathsf{KEM}}$.

BEGIN RESTATED

In game $\mathbf{G}_7$, whenever there is a flow 2 query where the message was honestly generated by a matching session, we do not decapsulate to obtain the pre-key. Instead, if the message was generated

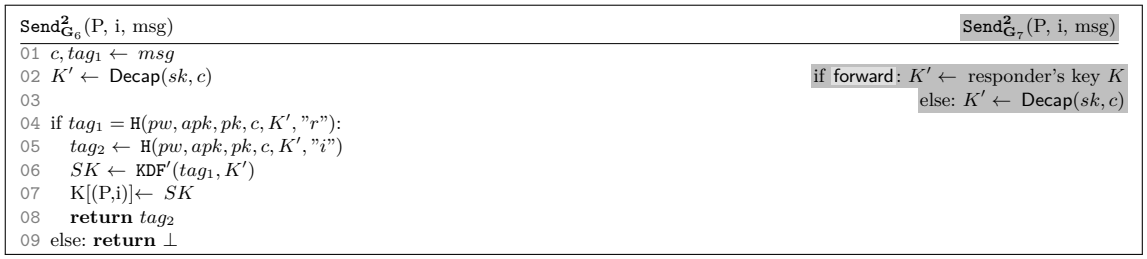


Fig. 16. In game $\mathbf{G}_7$, whenever forward occurs (i.e., c is a matching responder's honest ciphertext) we use the responder's pre-key K instead of decapsulating c.

by a matching session, we use the pre-key generated by that instance. Adversarially generated messages as well as ones that are forwarded from a non-matching session are decapsulated as before. This step is done in preparation for the reductions in the following two game hops. Games $\mathbf{G}_6$ and $\mathbf{G}_7$ are indistinguishable unless a correctness error occurred in game $\mathbf{G}_6$ and therefore $|\Pr[\mathtt{corrPW}_{\mathbf{G}_6}] - \Pr[\mathtt{corrPW}_{\mathbf{G}_7}]| = |\mathbf{Adv}_6 - \mathbf{Adv}_7| = n_a \cdot \delta^{\mathsf{KEM}}_{corr}$.

END RESTATED

Game $\boldsymbol{G}_8$: Abort on tag under correct pw. We now abort whenever the adversary actively interacts with a session, meaning the session is not the execution of two honest parties and submits a tag that was generated using the correct password. This models the case that the attacker successfully guessed the password and used it to attack an honest part, which can happen both for

1. initiators, in which case the attacker sent a responder tag *tag* that uses the correct password, and for
2. responders, in which case the attacker sent an *apk* that uses the correct password.

Here we focus on case 1, since the other case is handled by simulation SIM (see 3). We need to adapt previous classical reasoning about this change to capture that oracle H now is quantum-accessible – previous reasoning exploited that game and reductions can observe the issued queries to H, thus being able to immediately notice that *tag* was generated with the correct password. As a quantum counterpart to this approach, we use the extractable QROM formalism [DFMS22].

This models quantum-accessible random oracles like H as an extractable compressed random oracle eCO that has two interfaces, random oracle interface eCO.RO and an extraction interface $\mathtt{eCO.E}_f$ that is defined relative to a function $f : X \times Y = T$, where X and Y are the random oracle's domain and co-domain, respectively, and T is some other set. For our purposes, we identify X with the domain of H, so $X := \mathcal{K} \times \mathcal{D} \times \mathcal{PK}^2 \times \mathcal{C}$, and $Y := T := \mathcal{T} \times \mathcal{SK}$, the tag and session key space. Since we want to isolate the first part of the output, we choose the function f defined by the projection onto the tag space $f(input, y = (tag, K)) = tag$. Extraction interface $\mathtt{eCO.E}_f$ takes as input a classical value $t \in T$. It does a quantum analogue to going through the random oracle queries and returning an x such that $f(x, \mathtt{H}(x)) = t$: it performs suitable measurements that collapse the oracle database, just enough so that the classical procedure would yield one particular outcome x for all parts of the superposition. For our choice of f, $\mathtt{eCO.E}_f(tag)$ simply returns a random oracle pre-image of $tag||sk$ for any sk.

Equipped with this extraction interface, the game waits until the attacker has finished and then calls $\mathtt{eCO.E}_f$ on all 'dishonest' tries, i.e., all tags *tag* that were received by an honest initiator whose password was not corrupted, without *tag* having been computed during a preceding honest

execution of the respective responder. The game aborts if there exists a dishonest tag such that $\mathtt{eCO.E}_f(tag)$ returns a pre-image (K, pw, apk, pk', c) such that (K, apk, pk', c) matches the values which the initiator computed during the session to which the tag belongs. Since we already ruled out random oracle collisions, this in particular means that pw is the correct password. We denote this event by corrPW. We note that the extraction queries themselves do not change $\mathcal{A}$'s view since they are performed only after $\mathcal{A}$ finished [DFMS22, Th. 4.3, item 1].
We can now conclude that

$$\mathbf{Adv}_8 \leq \Pr[\,\mathsf{corrPW}\,]_{\mathbf{G}_8} \ .$$

Like in previous (classical) proofs, the change of probability for corrPW can be traced through the subsequent games $\mathbf{G}_8$-$\mathbf{G}_{14}$, by building anonymity and indistinguishability reductions that define their output based on corrPW.

Game $\mathbf{G}_9$: Make $\mathtt{eCO}$ extractions online. In this game, we switch to online extraction of the oracle used to derive the tag and session key. Instead of performing the extraction queries determining corrPW after running the attacker, we perform them already during the session runs. Whenever the attacker sends a tag tag to an honest initiator whose password was not corrupted, without tag having been computed during a preceding honest execution of the respective responder. This change is in preparation for the following two game hops, where the reductions need to detect certain edge-cases of adversarial input during the game.

To argue that this introduces little change, we use [DFMS22, Th. 4.3]: by properties 2.b and 2.c of Th. 4.3, any two subsequent queries to $\mathtt{eCO.RO}$ and $\mathtt{eCO.E}_f$ $8\sqrt{2\Gamma(f)/|\mathcal{K}| \cdot |\mathcal{T}|}$-almost-commute, where

$$\Gamma(f) = \max_t \max_x |\{y \mid f(x,y) = t\}|, \tag{3}$$

which for our choice $f(x,(tag,K)) = tag$ equals $|\mathcal{K}|$. This means that we can commute the extraction queries into the game to right after when the respective Send query was received. We need to commute at most $n_a \cdot (n_s + q_{\mathtt{H}}H)$ many times until the extractions are at the right place, thus $\mathbf{Adv}_9 \leq n_a \cdot (n_s + q_{\mathtt{H}})\sqrt{2/|\mathcal{T}|}$ and also $|\Pr\left[\,\mathsf{corrPW}\,_{\mathbf{G}_8}\right] - \Pr\left[\,\mathsf{corrPW}\,_{\mathbf{G}_9}\right]| \leq n_a \cdot (n_s + q_{\mathtt{H}})\sqrt{2/|\mathcal{T}|}$.

Game $\mathbf{G}_{10}$: Randomize public-key pk. In this game, we randomize the public key used to create the responder ciphertext. First we recall the setting: in order to unlink the first (apk) and second (c, tag) protocol message, we argue that the ciphertext c does not leak which ciphertext was used to create it. More formally, we replace the public key used to derive c with a fresh uniformly random one, and reduce the distance in the games induced by this change to the $\mathsf{ANO\text{-}PCA}$ security of KEM.
We give a reduction that simulates the PAKE game for an adversary $\mathcal{A}$ that distinguishes BPR game $\mathbf{G}_{10}$ from $\mathbf{G}_8$ and uses $\mathcal{A}$ to solve its $\mathsf{ANO\text{-}PCA}$ challenge. Consider the different query patterns that the reduction has to address. First we note that sessions that are passively attacked (queries to $\mathtt{Execute}$) or those where the initiator is honest can always be simulated without the use of programming in the ideal cipher: honest parties simply use the challenge public keys in place of their honest keys.
The more involved case is the one where the responder is honest, but the initiator is not: this setting involves an *adversarially chosen* encrypted public key apk that is decrypted by the honest responder into a public key. In order to move to game $\mathbf{G}_{10}$, where ciphertexts c are independent of the real public keys (or more accurately, the real password), the reduction reprograms decrypted public keys to challenge public keys in the $\mathsf{ANO\text{-}PCA}$ game.
We now show this proof step using only reprogramming according to the capabilities of SIM described above.

In a first step, the reduction answers all queries to E and D using the respective oracles of SIM. This change to SIM is not perfectly indistinguishable, therefore we have $\mathbf{Adv}_{10,1} \leq \varepsilon_{\mathsf{SIM}}$ where $\varepsilon_{\mathsf{SIM}}$ by definition of SIM.

The challenge set $\mathcal{X}$ used to initiate SIM is equal to the first set of public keys $\{pk_{0,0}, pk_{0,1}, \ldots pk_{0,n_a}\}$ provided by the anonymity challenger. This is the set that indicates the 'real' public keys, while the set $\{pk_{1,0}, pk_{1,1}, \ldots pk_{1,n_a}\}$ indicates the independent keys used *after* the reduction. Now, whenever the adversary submits a value apk for a session associated with password pw and key $k \leftarrow \mathtt{KDF}(pw)$, apk decrypts to a challenge public key whenever SIM programs successfully For $n_a$actively attacked sessions, we have by definition of SIM that

$$\Pr[(\vec{apk}, \vec{sid}) \leftarrow \mathcal{A}^{\mathsf{SIM}(\mathcal{X})} : \mathsf{D}_{\mathsf{SIM}}(k, apk_i) \in \mathcal{X} \forall (sid_i, apk_i) \in (\vec{sid}, \vec{apk_i})] = 1 - \delta_{\mathsf{SIM}}.$$

The reduction, on input apk, therefore decrypts apk to some challenge public key $pk_{0,j} \in \mathcal{X}$ and queries its challenger for a fresh anonymity challenge c related to that public key, and uses that in its response. That way, the adversary receives a ciphertext matching either pk_0 or pk_1 in each session, exactly as in the two games.

Finally, we note that since the reduction simulates all of $\mathcal{A}$'s oracles, so there is an edge case to consider: the case where $\mathcal{A}$ corrupts a party after it has queried the $\mathtt{Send}^0$ and $\mathtt{Send}^1$, but before it queries $\mathtt{Send}^2$ for matching sessions while forwarding messages. In this case, the oracle $\mathtt{Send}^2$ uses a challenge public key pk_0 that the attacker now *knows* due to the corruption. However, the reduction does not have the secret key associated with the public key, and it cannot decapsulate the ciphertext and recompute the tag. Therefore, when the oracle decides whether or not to reject an adversarially generated (c, tag), the reduction has use the extraction interface of the extractable random oracle $\mathtt{eCO}$ to determine the pre-key that was used by the adversary. Then, the reduction can query its plaintext-checking oracle to decide if the message should be rejected or not. Since we switched to the extractable oracle in a previous game, this does not incur any additional loss here.

Finally, the reduction outputs 0 whenever $\mathcal{A}$ wins, and 1 else. Therefore, the reduction wins whenever $\mathcal{A}$ wins and SIM successfully programmed for all its queries and we have $\mathbf{Adv}_{10,2} \leq \frac{1}{(1-\delta_{\mathsf{SIM}})} \cdot \mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}}$. We then arrive at the bounds

$$\mathbf{Adv}_{10} \leq \mathbf{Adv}_{10,1} + \mathbf{Adv}_{10,2} \leq \frac{1}{(1-\delta_{\mathsf{SIM}})} \cdot \mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}} + \varepsilon_{\mathsf{SIM}}.$$

$$|\Pr\left[\mathsf{corrPW}_{\mathbf{G}_9}\right] - \Pr\left[\mathsf{corrPW}_{\mathbf{G}_{10}}\right]| \leq \frac{1}{(1-\delta_{\mathsf{SIM}})} \cdot \mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}} + \varepsilon_{\mathsf{SIM}}.$$

Game $\boldsymbol{G}_{11}$: Randomize pre-key K. As previously, $\mathbf{Adv}_{11} \leq n_s \cdot (n_a + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM}}$.

BEGIN RESTATED

For all queries to the $\mathtt{Send}$ or $\mathtt{Execute}$ oracles where flag $\mathtt{trivGuess}$ is not raised before the query, we now randomize the pre-key K that is used to derive the final session key and the responder tag. For more details, see the pseudo-code in Figure 17. This change makes the pre-key independent of the ciphertext and the password for all fresh sessions. We now argue that an adversary noticing this change can be used to attack the indistinguishability property of the KEM. We define adversary $\mathcal{B}_0^{\mathsf{IND}}$ against the $\mathsf{IND\text{-}CPA}_{n,q_C}$ experiment (defined in Fig. 3) as follows (for the sake of formality, we give the pseudo-code of $\mathcal{B}_0^{\mathsf{IND}}$ in Fig. 18):

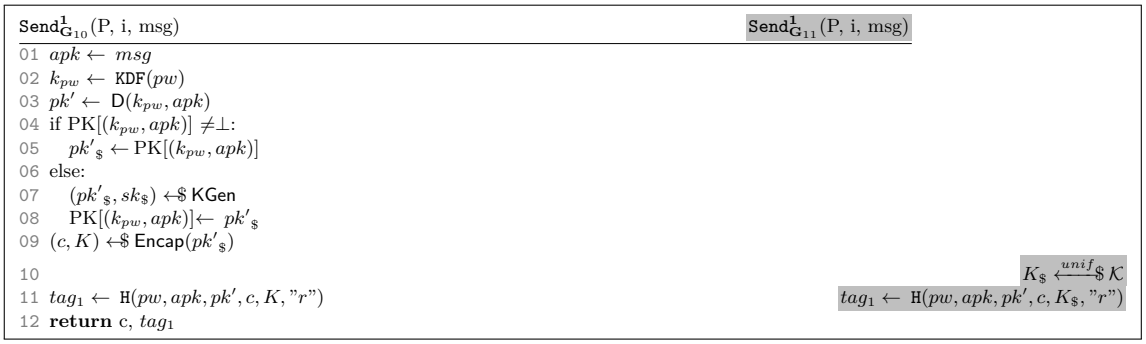


Fig. 17. In game $\mathbf{G}_{11}$, the pre-key set after querying `Send` or `Execute` is sampled independently of the password and the previous messages. Due to the change in game $\mathbf{G}_7$, this also randomizes the initiator side and we also write $K'_{\$} \leftarrow K_{\$}$.

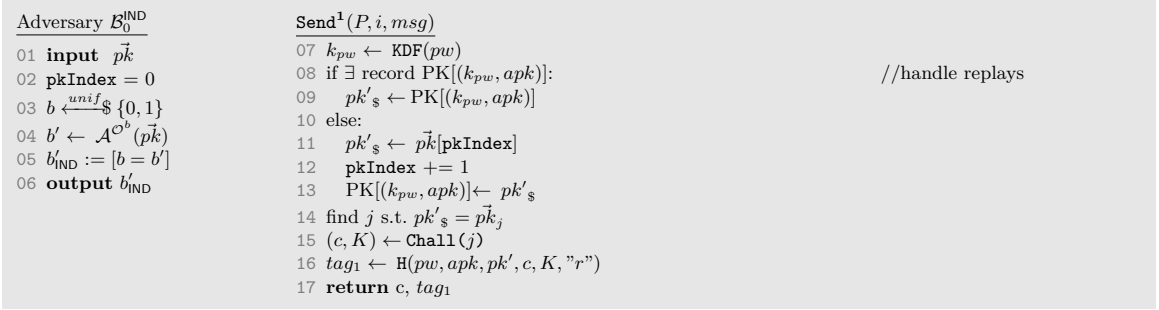


Fig. 18. $\mathsf{IND\text{-}CPA}_{n,q_C}$ adversary $\mathcal{B}_0^{\mathsf{IND}}$, used to reason about the hop from game $\mathbf{G}_{10}$ to $\mathbf{G}_{11}$. The set of oracles is $\mathcal{O} = \{\texttt{KDF}, \texttt{KDF'}, \mathsf{E}, \mathsf{D}, \texttt{Execute}, \texttt{Send}, \texttt{Reveal}, \texttt{Corrupt}\}$.

$\mathcal{B}_0^{\mathsf{IND}}$ receives a vector of challenge public keys $\vec{pk}$, of dimension $n = n_s$ and can query its challenge oracle `Chall`, provided by its $\mathsf{IND\text{-}CPA}_{n,q_C}$ challenger, at most $q_C = n_a + 1$ many times. $\mathcal{B}_0^{\mathsf{IND}}$ samples a challenge bit b', runs $\mathcal{A}$ and answers $\mathcal{A}$'s queries to the Oracles `H`, `E`, `KDF`, `Reveal`, $\texttt{Send}^{\mathbf{0}}$,$\texttt{Send}^{\mathbf{2}}$, and $\texttt{Test}^{b'}$ according to the oracles in $\mathbf{G}_{10}$.
On $\texttt{Send}^{\mathbf{1}}$ queries, $\mathcal{B}_0^{\mathsf{IND}}$ issues a $\texttt{Chall}(j)$ query to its own challenger receive (c^*, K^*), where j is the index of the public key which it uses to answer the query. If $\mathsf{trivGuess}$ has been raised, $\mathcal{B}_0^{\mathsf{IND}}$ generates a key pair and continues the protocol honestly without inserting any challenges in this session. If the same apk is submitted multiple times for sessions using the same password, the game is kept consistent by re-using the respective public key. When $\mathcal{A}$ outputs a guess b, $\mathcal{B}_0^{\mathsf{IND}}$ checks if $b = b'$. In the case that $b = b'$, it returns 1 as its own output bit, otherwise, it returns 0.

$\mathcal{B}_0^{\mathsf{IND}}$ perfectly simulates $\mathbf{G}_{10}$ when run in the $\mathsf{IND\text{-}CPA}_{n_s,n_a+1}$- game with challenge bit 0, $\mathbf{G}_{11}$ when run with challenge bit 1, and returns 1 if the adversary wins. Therefore, the difference between $\mathcal{A}$'s winning probabilities in games $\mathbf{G}_{10}$ and $\mathbf{G}_{11}$ is upper bounded by the respective $\mathsf{IND\text{-}CPA}_{n_s,n_a+1}$ advantage of $\mathcal{B}_0^{\mathsf{IND}}$ against KEM:

$$|\mathbf{Adv}_{10} - \mathbf{Adv}_{11}| \leq n_s \cdot (n_a + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM}}(\mathcal{B}_0^{\mathsf{IND}})$$

To keep track of the change in the probability of $\Pr[\texttt{corrPW}]$, we can adapt the reduction $\mathcal{B}_0^{\mathsf{IND}}$ exactly like in the game-hop before by redefining the output bit to be 1 iff `corrPW` occurred and $|\Pr[\texttt{corrPW}_{\mathbf{G}_{10}}] - \Pr[\texttt{corrPW}_{\mathbf{G}_{11}}]| \leq n_s \cdot (n_a + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM}}(\mathcal{B}_1^{\mathsf{IND}})$. At this point, pre-key K (for sessions between non-corrupted parties) is independent of the password and the protocol messages.

END RESTATED

Game $\boldsymbol{G}_{12}$: Move `eCO` extractions to end of game. In this game, we switch back to offline extraction, i.e., we only perform the extraction queries determining if a tag was valid after running the attacker. This is done in preparation for the next step that relies on measuring the compressed database at the end of the game. With the same reasoning as for game $\mathbf{G}_9$, we find

$$\mathbf{Adv}_{12} \leq n_a \cdot (n_s + q_{\mathtt{H}})\sqrt{2/|\mathcal{T}|}$$

and

$$|\Pr\left[\mathsf{corrPW}_{\mathbf{G}_{11}}\right] - \Pr\left[\mathsf{corrPW}_{\mathbf{G}_{12}}\right]| \leq n_a \cdot (n_s + q_{\mathtt{H}})\sqrt{2/|\mathcal{T}|} \ .$$

Game $\boldsymbol{G}_{13}$: Randomize Tag & Session Key. In the penultimate step of the proof, we randomize the responder tag tag and the session key SK, i.e., we simply sample random values from $\mathcal{T}$ and $\mathcal{SK}$, and return them as tag and session key. Recall, that in Game $\mathbf{G}_{12}$ the tag and session key are computed as $(tag, SK) \leftarrow \mathtt{H}(K_{\$}; pw, apk, pk', c)$ where $K_{\$}$ is a fresh uniformly random sample from $\mathcal{K}$ that is not used anywhere else in the protocol. Hence, the values in both games follow the same marginal distribution. However, in Game $\mathbf{G}_{12}$ they are consistent with the random oracle, while in Game $\mathbf{G}_{13}$ they are not.

Intuitively, an adversary can only notice this inconsistency in Game $\mathbf{G}_{13}$ if it ever learned the value of `H` at position $x^* = (K_{\$}; pw, apk, pk', c)$. Given that we simulate `H` using the compressed oracle technique, this means that the difference between this and the last game can be bounded by the probability that the final database D of the compressed oracle in Game $\mathbf{G}_{13}$ contains a value for x^* when measured in the end, i.e.,

$$\mathbf{Adv}_{13} \leq \Pr[D(x^*) \neq \bot].$$

It remains to quantify $\Pr[D(x^*) \neq \perp]$. For this, note that in Game $\mathbf{G}_{13}$, $K_\$$ is not used anywhere in the protocol anymore and therefore the whole view of the adversary is independent of $K_\$$. For that reason, we can delay the sampling of $K_\$$ until after we measured the database D. The database D has $q_\mathtt{H}$ entries. The probability that one of them is of the form $(K_\$; pw, apk, pk', c)$ for any value of $K_\$$ is therefore upper bounded by $1/|\mathcal{K}|$ and therefore

$$\mathbf{Adv}_{13} \leq \Pr[D(x^*) \neq \perp] \leq \frac{q_\mathtt{H}}{|\mathcal{K}|}.$$

Game $\boldsymbol{G}_{14}$: Randomize Passwords. It remains to upper-bound corrPW. In this game, the attacker's view is completely independent of the chosen passwords, up to corrupted ones. This means that we can replace all non-corrupted passwords with fresh ones at the end, after running the attacker, and defining event corrPW with respect to the resampled ones. We note that there exists exactly one pre-image (K, pw, apk, pk', c) per tag that could trigger corrPW, and that the simulation SIM does not raise this flag as it did in the previous proof.

We can now bound the probability of corrPW in game $\mathbf{G}_8$, using the number of send queries and the password distribution. Assuming a uniform distribution on a password dictionary of size $|\mathcal{D}|$, and upper-bounding $\mathcal{A}$'s number of send queries by n_a,

$$\Pr[\,\mathsf{corrPW}_{\mathbf{G}_{14}}] \leq \frac{n_a}{|\mathcal{D}|}$$

And therefore $\Pr[\,\mathsf{corrPW}_{\mathbf{G}_8}] \leq \sum_{i=8}^{14} \Pr[\,\mathsf{corrPW}_{\mathbf{G}_i}]$.

Since the passwords and session keys are completely random from the adversary's view, we have that $|\Pr[\mathbf{G}_{14}(\mathcal{A}) \Rightarrow 1]| = \frac{1}{2}$. We can finally sum up the terms and state the bound:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{BPR}}_{\mathsf{OCAKE}}(\mathcal{A}) &\leq \sum_{i=0}^{14} |\Pr[\mathbf{G}_i(\mathcal{A}) \Rightarrow 1]| \\
&\leq \frac{n_a}{|\mathcal{D}|} + n_a^2 \cdot \eta_{\mathsf{KGen}} + \frac{|\mathcal{D}|^2}{|\mathcal{K}_{pw}|} + \frac{80e^2 q_\mathtt{H}^3 + 2}{|\mathcal{T}|} + n_a \cdot \delta^{\mathsf{KEM}}_{corr} + \\
&\quad + 4 \cdot n_a \cdot (n_s + q_\mathtt{H})\sqrt{2/|\mathcal{T}|} + 2 \cdot (\frac{1}{(1-\delta_{\mathsf{SIM}})} \cdot \mathbf{Adv}^{\mathsf{ANO\text{-}PCA}}_{\mathsf{KEM}} + \varepsilon_{\mathsf{SIM}}) \\
&\quad + 2 \cdot n_s \cdot (n_a + 1) \cdot \mathbf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{KEM}} + 2 \cdot \frac{q_\mathtt{H}}{|\mathcal{K}|}
\end{aligned}$$

B Notes on (O)CAKE security and the ideal cipher model

To give intuition for the needed properties of SIM, this section recollects the various generic attacks on PAKE protocols, how these attacks inflict conditions on OCAKE's building blocks, and the proof techniques used to capture these conditions when idealizing the block cipher as an ideal cipher.

Generic attacks on (O)CAKE and their relation to the ideal cipher

We now recollect generic attacks on PAKE protocols, how they relate to OCAKE's building blocks, and the usage of the ideal cipher in security proofs.

Initialisation

01 for $(P, P') \in \mathcal{P} \times \mathcal{P}$
02 $pw \leftarrow\$ \mathrm{PW}()$
03 $\mathrm{PWD}[\{P, P'\}] \leftarrow pw$

$\underline{\texttt{Execute}(P, i, P', j)}$

04 $apk \leftarrow \texttt{Send}^{\mathbf{0}}(P, i, \bot)$
05 $c, tag \leftarrow \texttt{Send}^{\mathbf{1}}(P', j, apk)$
06 $\texttt{Send}^{\mathbf{2}}(P, i, (c, tag))$
07 MANIP[{P,:}]$\leftarrow$ `false`
08 **return** (apk, c, tag_2, tag_1)

$\underline{\texttt{Corrupt}(P, \mathrm{PWD'})}$

09 $\mathrm{PWD}_P \leftarrow \mathrm{PWD}[\{P, :\}]$
10 CRPT[{P,:}]$\leftarrow$ `true`
11 $\mathrm{PWD}[\{P, :\}] \leftarrow \mathrm{PWD'}[\{P, :\}]$
12 **return** PWD_P

$\underline{\texttt{Reveal}(P, i)}$

13 $\mathrm{RVL}[(P, i)] \leftarrow$ `true`
14 **return** $\mathrm{SK}[(P, i)]$

$\underline{\texttt{Test}^b(P, i)}$

15 $SK_0 \leftarrow \mathrm{K}[(P, i)]$
16 $SK_1 \xleftarrow{unif}\$ \mathcal{SK}$
17 if (CRPT[{P,P'}] and MANIP$[(P, i)]$)
18 or if (RVL$[(P, i)]$ or RVL$[(P', j)]$)
19 or if (K$[(P, i)] = \bot$): **return** $\bot$
20 else: **return** SK_b

$\underline{\texttt{Send}^{\mathbf{0}}(P, i, msg)}$

21 MANIP$[(P, i)]\leftarrow$ `true`
22 $k_{pw} \leftarrow \texttt{KDF}(pw)$
23 $(pk, sk) \leftarrow\$ \mathsf{KGen}$
24 $apk \leftarrow \mathsf{E}(k_{pw}, pk)$
25 **return** apk

$\underline{\texttt{Send}^{\mathbf{1}}(P, i, msg)}$

26 MANIP$[(P, i)]\leftarrow$ `true`
27 $apk \xleftarrow{parse} msg$
28 $k_{pw} \leftarrow \texttt{KDF}(pw)$
29 $pk' \leftarrow \mathsf{D}(k_{pw}, apk)$
30 $(c, K) \leftarrow\$ \mathsf{Encap}(pk')$
31 $tag, SK \leftarrow \mathtt{H}(K, pw, apk, pk', c)$
32 $\mathrm{SK}[(P, i)] \leftarrow SK$
33 **return** (c, tag)

$\underline{\texttt{Send}^{\mathbf{2}}(P, i, msg)}$

34 MANIP$[(P, i)]\leftarrow$ `true`
35 $c, tag_1 \xleftarrow{parse} msg$
36 $K' \leftarrow \mathsf{Decap}(sk, c)$
37 $tag', SK' \leftarrow \mathtt{H}(K'; pw, apk, pk, c)$
38 `if` $tag' = tag$
39 $\mathrm{SK}[(P, i)] \leftarrow SK$

Fig. 19. The oracles in the security game for OCAKE. PWD is the dictionary of the parties' passwords and CRPT, MANIP and RVL indicate the corruption status of a session. Password generation in the initialization phase (**Initialization**) is modeled using the long-lived key generator PW.

Initialisation

01 for $(P, P') \in \mathcal{P} \times \mathcal{P}$
02 $pw \leftarrow\$ \mathrm{PW}()$
03 $\mathrm{PWD}[\{P, P'\}] \leftarrow pw$
04 $PK \leftarrow \emptyset$

$\underline{\texttt{Send}^{\mathbf{0}}(P, i, msg)}$

05 MANIP$[(P, i)]\leftarrow$ `true`
06 $k_{pw} \leftarrow \texttt{KDF}(pw)$
07 $(pk, sk) \leftarrow\$ \mathsf{KGen}$
08 if $pk \in PK$ abort
09 $PK = PK \cup \{pk\}$
10 $apk \leftarrow \mathsf{E}(k_{pw}, pk)$
11 **return** apk

Fig. 20. The game change for game $\mathbf{G}_1$ Abort on KGen Collision.

Initialisation

01 for $(P, P') \in \mathcal{P} \times \mathcal{P}$
02 $pw \leftarrow\$ \mathrm{PW}()$
03 $\mathrm{PWD}[\{P, P'\}] \leftarrow pw$
04 $PK \leftarrow \emptyset$
05 $\mathcal{L}_{pw} \leftarrow \emptyset$

$\underline{\mathtt{Send}^{\mathbf{1}}(P, i, msg)}$

06 $\mathrm{MANIP}[(P, i)] \leftarrow \mathtt{true}$
07 $apk \xleftarrow{parse} msg$
08 $k_{pw} \leftarrow \mathtt{KDF}(pw)$
09 $pk' \leftarrow \mathsf{D}(k_{pw}, apk)$
10 $(c, K) \leftarrow\$ \mathsf{Encap}(pk')$
11 $tag, SK \leftarrow \mathtt{H}(K, pw, apk, pk', c)$
12 $\mathrm{SK}[(P, i)] \leftarrow SK$
13 **return** (c, tag)

$\underline{\mathtt{Send}^{\mathbf{0}}(P, i, msg)}$

14 $\mathrm{MANIP}[(P, i)] \leftarrow \mathtt{true}$
15 $k_{pw} \leftarrow \mathtt{KDF}(pw)$
16 if $k_{pw} \in \mathcal{L}_{pw}$ abort
17 $\mathcal{L}_{pw} = \mathcal{L}_{pw} \cup \{k_{pw}\}$
18 $(pk, sk) \leftarrow\$ \mathsf{KGen}$
19 if $pk \in PK$ abort
20 $PK = PK \cup \{pk\}$
21 $apk \leftarrow \mathsf{E}(k_{pw}, pk)$
22 **return** apk

Fig. 21. The game change for game $\mathbf{G}_2$ Abort on KDF Collision.

Initialisation

01 for $(P, P') \in \mathcal{P} \times \mathcal{P}$
02 $pw \leftarrow\$ \mathrm{PW}()$
03 $\mathrm{PWD}[\{P, P'\}] \leftarrow pw$
04 $PK \leftarrow \emptyset$
05 $\mathcal{L}_{pw} \leftarrow \emptyset$
06 $C \leftarrow \emptyset$

$\underline{\mathtt{Send}^{\mathbf{1}}(P, i, msg)}$

07 $\mathrm{MANIP}[(P, i)] \leftarrow \mathtt{true}$
08 $apk \xleftarrow{parse} msg$
09 $k_{pw} \leftarrow \mathtt{KDF}(pw)$
10 $pk' \leftarrow \mathsf{D}(k_{pw}, apk)$
11 $(c, K) \leftarrow\$ \mathsf{Encap}(pk')$
12 $C \leftarrow C \cup c$
13 $tag, SK \leftarrow \mathtt{H}(K, pw, apk, pk', c)$
14 $TAG \leftarrow TAG \cup tag$
15 $\mathrm{SK}[(P, i)] \leftarrow SK$
16 **return** (c, tag)

$\underline{\mathtt{Send}^{\mathbf{2}}(P, i, msg)}$

17 $\mathrm{MANIP}[(P, i)] \leftarrow \mathtt{true}$
18 $c, tag_1 \xleftarrow{parse} msg$
19 $K' \leftarrow \mathsf{Decap}(sk, c)$
20 if $c \in C : K' \leftarrow K$
21 $tag', SK' \leftarrow \mathtt{H}(K'; pw, apk, pk, c)$
22 `if` $tag' = tag$
23 $\mathrm{SK}[(P, i)] \leftarrow SK$

Fig. 22. The game change for game $\mathbf{G}_7$ Do not decrypt honest c. Set the responder's pre-key to the matching initiator's.

Initialisation

01 for $(P, P') \in \mathcal{P} \times \mathcal{P}$
02 $pw \leftarrow\$ \mathrm{PW}()$
03 $\mathrm{PWD}[\{P, P'\}] \leftarrow pw$
04 $PK \leftarrow \emptyset$
05 $\mathcal{L}_{pw} \leftarrow \emptyset$
06 $C \leftarrow \emptyset$
07 $TAG \leftarrow \emptyset$

$\underline{\texttt{Send}^{\texttt{2}}(P, i, msg)}$

08 $\mathrm{MANIP}[(P, i)] \leftarrow \texttt{true}$
09 $c, tag_1 \xleftarrow{parse} msg$
10 $K' \leftarrow K$
11 $tag', SK' \leftarrow \mathtt{H}(K', pw, apk, pk, c)$
12 $\texttt{if } tag' = tag$
13 if not $tag \in TAG$ and not $\mathrm{CRPT}[\{\mathrm{P,P'}\}]$ abort
14 $\mathrm{SK}[(P, i)] \leftarrow SK$

Fig. 23. The game change for game $\mathbf{G}_8$ Abort on correct password. Abort if we receive a tag that is correct unless it is honest or the password was corrupted.

$\underline{\texttt{Send}^{\texttt{1}}(P, i, msg)}$

01 $\mathrm{MANIP}[(P, i)] \leftarrow \texttt{true}$
02 $apk \xleftarrow{parse} msg$
03 $k_{pw} \leftarrow \mathtt{KDF}(pw)$
04 $pk' \leftarrow \mathsf{D}(k_{pw}, apk)\, pk_{\$} \leftarrow \mathcal{PK}$
05 $(c, K) \leftarrow\$ \mathsf{Encap}(pk')\, (c, K) \leftarrow\$ \mathsf{Encap}(pk'_{\$})$
06 $C \leftarrow C \cup c$
07 $tag, SK \leftarrow \mathtt{H}(K, pw, apk, pk', c)$
08 $TAG \leftarrow TAG \cup tag$
09 $\mathrm{SK}[(P, i)] \leftarrow SK$
10 **return** (c, tag)

Fig. 24. The game change for game $\mathbf{G}_{10}$ Anonymity.

$\underline{\texttt{Send}^{\texttt{1}}(P, i, msg)}$

01 $\mathrm{MANIP}[(P, i)] \leftarrow \texttt{true}$
02 $apk \xleftarrow{parse} msg$
03 $k_{pw} \leftarrow \mathtt{KDF}(pw)$
04 $pk_{\$} \leftarrow \mathcal{PK}$
05 $(c, K) \leftarrow\$ \mathsf{Encap}(pk'_{\$})$
06 $K_{\$} \leftarrow\$ \mathcal{K}$
07 $C \leftarrow C \cup c$
08 $tag, SK \leftarrow \mathtt{H}(K, pw, apk, pk', c)\, tag, SK \leftarrow \mathtt{H}(K_{\$}, pw, apk, pk', c)$
09 $TAG \leftarrow TAG \cup tag$
10 $\mathrm{SK}[(P, i)] \leftarrow SK$
11 **return** (c, tag)

Fig. 25. The game change for game $\mathbf{G}_{11}$ Indistinguishability.

$\underline{\texttt{Send}^{\mathbf{1}}(P, i, msg)}$

01 MANIP$[(P,i)] \leftarrow$ `true`
02 $apk \xleftarrow{parse} msg$
03 $k_{pw} \leftarrow \texttt{KDF}(pw)$
04 $pk_{\$} \leftarrow \mathcal{PK}$
05 $(c, K) \leftarrow\$ \mathsf{Encap}(pk'_{\$})$
06 $C \leftarrow C \cup c$
07 $K_{\$} \leftarrow\$ \mathcal{K}$
08 $tag, SK \leftarrow \texttt{H}(K_{\$}, pw, apk, pk', c)$
09 $TAG \leftarrow TAG \cup tag$
10 $tag_{\$} \leftarrow\$ \mathcal{T}$,$SK_{\$} \leftarrow\$ \mathcal{SK}$
11 SK$[(P,i)] \leftarrow SK$ SK$[(P,i)] \leftarrow SK_{\$}$
12 **return** (c, tag) **return** $(c, tag_{\$})$

Fig. 26. The game change for game $\mathbf{G}_{13}$ Randomize Tag and Session key.

Dictionary attacks and anonymity The first type of attack on PAKE protocols, including OCAKE, is a dictionary attack. In a dictionary attack, an attacker observes the protocol message exchanged between honest parties and attempts to extract information about the password (i.e., the long-term secret) from the protocol messages. As the password is what authenticates the honest parties, any successful dictionary attack which recovers the password results in a complete loss of security. For OCAKE (and CAKE), it is thus crucial that the second protocol message does not reveal the public key pk that was sent in the first message: knowing pk would allow to test password guesses, by encrypting pk under the different passwords and comparing the result against apk. For OCAKE, the second protocol message is a KEM ciphertext, it is thus natural to require that KEM satisfies anonymity, i.e., that KEM ciphertexts do not indicate to which public key they belong. Accordingly, the known security proofs utilize a reduction to the anonymity of KEM. This reduction has to distinguish ciphertexts belonging to proper public keys from ciphertexts belonging to random keys, and tricks the attacker into solving this task by programming its challenge public keys into suitable positions of the ideal cipher. This shows up, e.g., in game $\mathbf{G}_9$ of the OCAKE security proof given in [AHH+23], and in games $\mathbf{G}_8$, $\mathbf{G}_9$ and $\mathbf{G}_{10}$ of the CAKE security proof given in [PZ23].

In section 5 we show an attack on the protocol based on a lack of anonymity.

Dictionary attacks and public-key uniformity Another dictionary attack vulnerability appears for OCAKE (and CAKE) if the message apk does not decrypt to a valid public key under every password, as attackers can then rule out the passwords for which decrypting apk would yield an invalid public key. This can be captured by requiring that the KEM additionally satisfies public-key uniformity, i.e., that the public output parts of KGen are indistinguishable from random elements in the range of BC^{-1}. Similar to the anonymity reduction, the uniformity reduction will trick the attacker into solving the task at hand by programming its challenge public keys into suitable positions of the ideal cipher.

Password guessing attacks and extraction Since the passwords are assumed to be low-entropy, it is possible for attackers to guess the password and use their guess to attempt at a successful interaction with (or 'online' attack on) the honest party. Security proofs bound the probability of this event based on the dictionary size and leakage from previous sessions. This requires, however, that the involved security reductions can efficiently detect this event - the reductions are used to randomize certain outputs, but only if no password guess occurs, since in that case the randomization would be noticeable by the adversary. This means that we need a way to *extract* the password that is associated to the adversary's message.

Collision attacks The security of PAKE protocols significantly decreases if attackers can use a single interaction (or 'online' attack) with an honest party to rule out several passwords from the (already comparably small) list of passwords. Accordingly, the next attack vector we consider are collision attacks, i.e., attacks that leverage the fact that an adversarially chosen protocol message would work for several distinct passwords. The proofs therefore will argue at some point that the attacker cannot come up with a message that could be used to rule out or confirm more than 1 password. (An equivalent requirement holds for UC PAKE). For OCAKE, an adversary could find an apk value s.th. apk is a valid encryption for two distinct passwords and public keys chosen by the adversary $(pw, pk), (pw', pk')$, i.e., such that we have $apk = \mathsf{BC}(pw, pk) = \mathsf{BC}(pw', pk')$, meaning the adversary might be able to try both passwords at once. Modeling BC as an ideal cipher and the derivation functions as random oracles, the proof of OCAKE accordingly eliminates/bounds the probability of certain collision events, as seen in games $\mathbf{G}_0$ to $\mathbf{G}_5$ [AHH+23]. Looking ahead, we will have to find equivalent counterparts to these arguments in the setting where the involved oracles are quantum-accessible. We have shown that the last remaining piece to the security proof of OCAKE considering quantum attackers is the reduction towards anonymity. In the classical proof, this step makes use of the ideal cipher model. However, when proving security against quantum attackers, we have to model the block cipher in a way that lets an attacker query it on arbitrary quantum states. To the best of our knowledge, there is currently no way to argue simulation of a quantumly accessible block cipher that allows for proof techniques such as reprogramming, lazy sampling of extraction in an equivalent way. Indeed, it may prove impossible to achieve these properties. However, it was never shown that using programmability is the only way to prove the statement. Hence, we are taking a step back and try and figure out the exact requirements on a simulator for the ideal cipher (and if we need one at all). To this end, we first provide more technical details and outline properties of the ideal cipher that play an important role in the existing proofs of both OCAKE (B) and CAKE (B). This analyses also allows us to determine some lower bounds on the parameters of SIM.

Role of the ideal cipher in the proof of OCAKE In this section, we recall the important role played by the ideal cipher in the proof of OCAKE that was given in [AHH+23]. Intuitively, the relevant proof steps have the following purpose: they unlink the first and second message in the protocol to prevent dictionary attacks on the password. In the proof of OCAKE, this is reflected as a reduction to the anonymity property of KEM, performed in game $\mathbf{G}_9$. We will now briefly describe the role of the ideal cipher in that part of the proof.

In order to be successful, the anonymity reduction simulates the game to the attacker in a way that suitably places its challenges: it needs to make sure that (1) the ciphertexts used by the (simulated) responder are its challenge ciphertexts and (2) these ciphertexts are consistent with the protocol execution. It is convenient to argue separately about the case where the attack is on the responder or initiator side of the protocol. I.e., when the attacker is on the responder side, the reduction simulates the initiator oracle, and vice versa.

When simulating the initiator oracle, the reduction uses a challenge public key provided by the anonymity challenger and writes it into the ideal cipher simulation. This step is equivalent to querying IC encryption on that public key.

Conversely, when simulating a responder oracle, the reduction uses a challenge ciphertext whenever apk decrypts to one of its challenge public keys. An arising subtlety, however, is that the received apk might not have originated from the initiator oracle, but rather be a value that the attacker chose. This is problematic because the simulated responder needs to answer its query according to the honest decryption of this apk. Therefore, if the reduction wants to answer the query with a challenge ciphertext, the decryption must correspond to a challenge public key.

In the classical proof, the reduction simply answers all ideal cipher decryption queries (made by either adversary or game) with challenge public keys, unless the requested apk originates from a

previous query in the forwards direction, i.e., a query to IC for some pk under the correct password of that session. This is shown schematically in Fig. 27. In consequence, D will return a challenge public key, thus circumventing the issue of not being able to output a challenge ciphertext in every session, unless the adversary made a forwards query with the correct password. In the proof of OCAKE, the probability of an attacker making such a forwards query is bounded by the probability of the attacker *guessing* the password of some session. There is one more subtlety to this step: in

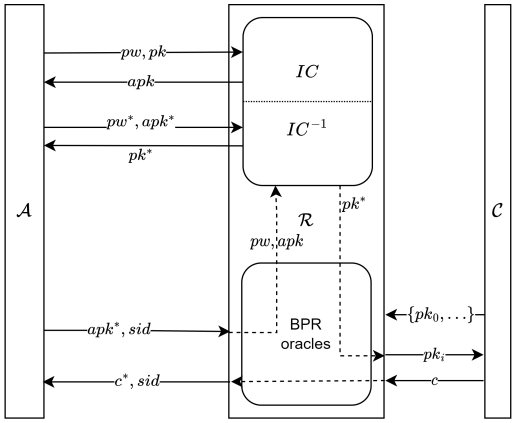


Fig. 27. Schematic view of anonymity reduction $\mathcal{R}$, used to decouple KEM ciphertexts from their originating public keys. $\mathcal{R}$ receives challenge public keys and ciphertexts from ANO-PCA challenger $\mathcal{C}$ and interacts with the PAKE adversary $\mathcal{A}$ to solve its challenges. $\mathcal{R}$ provides the PAKE BPR oracles and the ideal cipher oracles.
When receiving a message apk from $\mathcal{A}$ whose decryption pk^* equals a challenge public key pk_i, $\mathcal{R}$ responds with an anonymity challenge c. $\mathcal{A}$ can make arbitrary quantum queries to IC at any time. (shown in red).

the case that public keys of KEM are not uniformly random elements of the message space $\mathcal{M}$ of the cipher, reprogramming outputs changes the output distribution. To address this issue, the proof adds a preliminary step that changes all decryption output samplings to be outputs of the key generation KGen in proof step $\mathbf{G}_6$. This step reduces the distance of the change to the public key uniformity PKU of KEM.

Take-away: exploited IC properties To summarize, the proof exploited the following properties of the ideal cipher model: a reduction simulating IC can

- observe and keep a list of all queries issued by the attacker, thereby being able to extract the information that a correct password guess occurred from the received apk; and
- embed challenges into the ideal cipher outputs when it suits its goals, circumventing potential inconsistencies by using the query list to account for previous IC queries.

Role of the ideal cipher in the proof of CAKE We now also briefly describe the ways in which the CAKE proof given in [PZ23] relies on similar properties of the two involved ideal ciphers (E_1,D_1,E_2,D_2). There are four game-hops that require embedding challenge values into the ideal ciphers' decryption outputs:

- Game $\mathbf{G}_2$: Decryption queries to the first cipher D_1 are sampled using KEM key generation (KEM fuzziness/Public-key uniformity)
- Game $\mathbf{G}_8$: $\mathcal{B}_4$ embeds challenge ciphertexts c into queries to D_2 (OW-rPCA security of KEM)

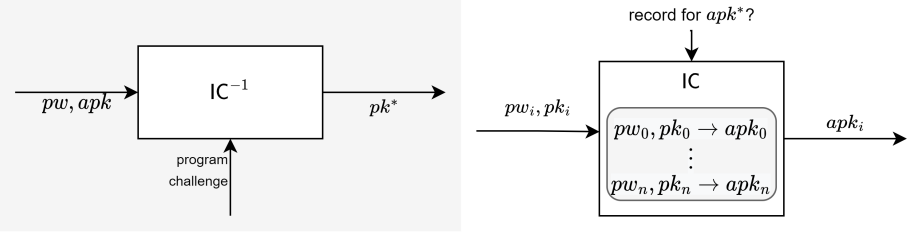


Fig. 28. Proof techniques used in the existing OCAKE proof: (r) In order to insert its anonymity challenges into the communication, the anonymity reduction sets all IC decryption outputs to challenge public keys. (l) Identifying correct password guesses (raising flag corrPW) requires checking if an (adversarially chosen) *apk* resulted from an (adversarial) IC encryption query that used the correct password.

- Game $\mathbf{G}_9$: $\mathcal{B}_5$ embeds challenge public keys pk into queries to D_1 (OW-PCA security of KEM)
- Game $\mathbf{G}_{10}$: $\mathcal{B}_6$ embeds challenge public keys pk into queries to D_1 (ANO-PCA security of KEM)

Two of these cases are conceptually equivalent to steps in the security proof for OCAKE. (Game $\mathbf{G}_2$ for CAKE conceptually resembles game $\mathbf{G}_6$ in the proof of OCAKE, and game $\mathbf{G}_{10}$ for CAKE conceptually resembles OCAKE game $\mathbf{G}_7$). The other two games pose additional requirements: Game $\mathbf{G}_8$ concerns the second ideal cipher that is used in the CAKE protocol, and does therefore not have a direct equivalent. The change in game $\mathbf{G}_9$ of CAKE is conceptually equivalent to the change in game $\mathbf{G}_{10}$ of OCAKE. Notably though, this step requires programmability, whereas the equivalent step in OCAKE ($\mathbf{G}_{10}$ towards IND-CPA) does not due to its alternative proof structure. For OCAKE, the proof uses the anonymity of KEM before one-wayness or indistinguishability.

This difference also results in a difference in the requirements on KEM: OCAKE relies on the stronger notion of ANO-PCA security, while CAKE only requires weak anonymity. To summarize the difference, strong anonymity also holds for adversaries that have access to the key contained in the encapsulation. Since we aim to minimize the amount of programming done in the proof in this work, we will continue the analysis for the case of the OCAKE protocol.

This analyses also allows us to determine some lower bounds on the parameters of SIM.

Lower bounds on SIM parameters Now that we have shown that the simulation above is both necessary and sufficient to prove the security of OCAKE, we discuss its instantiation. When modeling assuming purely classical adversaries, we see that lazy sampling and reprogramming according to the ideal cipher model provides the necessary properties. As soon as we model quantum queries, these properties are no longer obvious. In this work we defined the minimum necessary properties as a first step to bridge the gap to a proof against quantum attackers. Since the new setting allows both quantum and classical attacks, we can use classical reasoning to define some lower bounds on the achievability of SIM.

We see that SIM is defined by two parameters, $\varepsilon_{\mathsf{SIM}}$ and δ_{SIM}. Informally, $\varepsilon_{\mathsf{SIM}}$ indicates the distance between the simulation and an ideal cipher, which can also be interpreted as the probability of distinguishing one from the other of an adversary interacting with one. An example would be the case where the simulation reprograms an already queried position: an adversary making two queries would detect this.

Parameter δ_{SIM} can be interpreted as the success probability of programming. This parameter will depend both on the instantiation of SIM as well as the setting where SIM is used: the adversary could output values that were the result of encryption queries. In those cases, the decrypted value is adversarially chosen (unless SIM reprograms the output). in the case of PAKE, this only happens whenever the attacker correctly guesses the password of a session, and we can upper-bound that probability. However, the probability of such an event still gives a lower bound on δ_{SIM} and $\varepsilon_{\mathsf{SIM}}$.

Intuitively, either δ_{SIM} or $\varepsilon_{\mathsf{SIM}}$ or some combination of the two must be larger than the probability of this event. In the case of OCAKE, the relation takes the form $\varepsilon + \delta \geq \Pr[\,\mathsf{corrPW}\,] = \frac{n_a}{|\mathcal{D}|}$ for n_a actively attacked sessions and a password dictionary $\mathcal{D}$.
Note that these lower bounds apply both in the classical and the quantum case.

Collisions in SIM and IC The final case to discuss is that of collisions: an ideal cipher $\mathsf{IC} : \mathcal{D} \times \mathcal{M} \to \mathcal{M}$ always contains points s.t. for two passwords $(pw, pw') \in \mathcal{D}^2$ and two messages $(m, m') \in \mathcal{M}^2$, we have that $\mathsf{E}(k_{pw}, m) = \mathsf{E}(pw', m)$. An adversary that can find such collisions can more easily trigger the bad case where the decryption is outside of the challenge set. Even without knowing which pw^* is used to decrypt, an adversary outputting this apk doubles the probability of the decryption being one of $\mathcal{A}$'s chosen values. This increase also poses a lower bound on the simulation parameters. In the case of OCAKE, the relation takes the form $\varepsilon + \delta \geq \Pr[\,\mathsf{corrPW}\,] + \Pr[\,\mathsf{collision}\,]$ for n_a actively attacked sessions and a password dictionary $\mathcal{D}$.