Group Action Key Encapsulation and Non-Interactive Key Exchange in the QROM

Julien Duman, Dominik Hartmann, Eike Kiltz,
Sabrina Kunzweiler, Jonas Lehmann, Doreen Riepel

Ruhr-Universität Bochum, Germany
{julien.duman,dominik.hartmann,eike.kiltz,
sabrina.kunzweiler,jonas.lehmann-c6j,doreen.riepel}@rub.de

Abstract. In the context of quantum-resistant cryptography, cryptographic group actions offer an abstraction of isogeny-based cryptography in the Commutative Supersingular Isogeny Diffie-Hellman (CSIDH) setting. In this work, we revisit the security of two previously proposed natural protocols: the Group Action Hashed ElGamal key encapsulation mechanism (GA-HEG KEM) and the Group Action Hashed Diffie-Hellman non-interactive key-exchange (GA-HDH NIKE) protocol. The latter protocol has already been considered to be used in practical protocols such as Post-Quantum WireGuard (S&P '21) and OPTLS (CCS '20).

We prove that *active* security of the two protocols in the Quantum Random Oracle Model (QROM) inherently relies on very strong variants of the Group Action Strong CDH problem, where the adversary is given arbitrary *quantum access* to a DDH oracle. That is, quantum accessible Strong CDH assumptions are not only sufficient but also necessary to prove active security of the GA-HEG KEM and the GA-HDH NIKE protocols.

Furthermore, we propose variants of the protocols with QROM security from the classical Strong CDH assumption, i.e., CDH with classical access to the DDH oracle. Our first variant uses key confirmation and can therefore only be applied in the KEM setting. Our second but considerably less efficient variant is based on the twinning technique by Cash et al. (EUROCRYPT '08) and in particular yields the first actively secure isogeny-based NIKE with QROM security from the standard CDH assumption.

Keywords: Group actions, CSIDH, Hashed ElGamal, NIKE, QROM, twinning

Table of Contents

1 Introduction

A non-interactive key exchange (NIKE) is a protocol that allows two parties to establish a common secret key in a non-interactive way. The first and most famous NIKE is the Diffie-Hellman key exchange [15] which forms the basis for a lot of other cryptographic protocols like ElGamal [18]. Most notably however, the existence of a secure NIKE implies secure key encapsulation mechanisms (KEM) (and hence public-key encryption) and authenticated key exchange (AKE) [20]. A NIKE can therefore be seen as one of the most basic and important primitives in cryptography.

The emergence of quantum computing however continues to have an unprecedented impact on public key cryptography. When scaled to a suitable size, quantum computers pose a threat to almost all classical public-key primitives, including Diffie-Hellman and ElGamal [37]. To mitigate this threat, researchers started building quantum resisting public-key cryptography based on certain quantum-hard problems on codes, lattices and isogenies. Even though quantum-resistant public-key encryption from lattices seems to offer the favorable trade-off over codes and isogenies in terms of speed, ciphertext expansion, and security, building an efficient (even passively secure) NIKE from codes or lattices remains an unsolved research problem.

Isogeny-based Cryptography. A promising alternative approach to post-quantum security is based on isogenies. An isogeny is a non-constant homomorphism between elliptic curves. In an algebraic context, isogenies can be used to build a commutative group action that behaves similarly to exponentiation in finite fields. This was first observed by Couveignes [14] and independently by Rostovtsev and Stolbunov [35]. The first practical instantiation was obtained by Castryck et al. [12] which in contrast to previous work uses the group action on the set of *supersingular* elliptic curves. Throughout this paper, we will use the abstract framework of cryptographic group actions introduced by Alamati et al. [2] to model isogeny-based constructions. (See Section 2.3 for formal definitions.) At a syntactical level, cryptographic group actions allow for a simple Group Action Diffie-Hellman ($\mathsf{GA\text{-}DH}$) key exchange and Group Action ElGamal ($\mathsf{GA\text{-}EG}$) public-key encryption scheme. With this abstraction in mind, the famous Commutative Supersingular Isogeny Diffie-Hellman (CSIDH) key exchange protocol of [12] can be seen as a specific instantiation of $\mathsf{GA\text{-}DH}$.

For cryptographic group actions, the analog of the traditional Computational Diffie-Hellman assumption (over prime-order groups) is the *Group Action Computational Diffie-Hellman assumption* ($\mathsf{GA\text{-}CDH}$) [14,35,12,2], see also Definition 6. $\mathsf{GA\text{-}CDH}$ is sufficient to prove passive security of "hashed versions" of $\mathsf{GA\text{-}DH}$ and $\mathsf{GA\text{-}EG}$ in the random oracle model. In analogy to the prime-order group setting, for active security one requires a "strong" type of Computational Diffie-Hellman assumption [1]. Providing the adversary additional access to a Group Action Decisional Diffie-Hellman oracle $\mathsf{GA\text{-}DDH}(\cdot,\cdot)$, i.e. an oracle which tells us whether a pair of elements forms a Diffie-Hellman tuple, defines the *Group Action Strong Computational Diffie-Hellman assumption* ($\mathsf{GA\text{-}StCDH}$). The prefix *strong* refers to the fact that the first input to this oracle is fixed (as opposed to the stronger and non-falsifiable *gap* assumptions). This assumption is well-known in the standard prime-order group setting and has already been used in proving active security of several protocols [28,29,39] in the group action setting as well.[1]

Quantum Random Oracle Model. The random-oracle model (ROM) [7] is commonly used in modern cryptography to argue *practical security* of cryptographic schemes. Adversaries with access to quantum computers will be able to implement the hash function on those, and therefore can evaluate the hash function on arbitrary quantum superpositions. To account for this gain in capabilities, the *quantum(-accessible)* random-oracle model (QROM) has been introduced [9]. The QROM has become the accepted model for proving post-quantum security and it is generally believed that proofs in the classical ROM are not sufficient to claim post-quantum security.

Actively secure KEMs and NIKE Protocols. In this work we are interested in constructing actively (i.e. IND-CCA) secure KEMs and actively secure NIKE protocols over cryptographic group actions.

Let us first look at the simpler case of KEMs. Generally speaking, we know of two natural approaches to build efficient IND-CCA secure KEMs. The first approach is generic and applies the Fujisaki-Okamoto

[1] We stress that $\mathsf{GA\text{-}StCDH}$ over standard cryptographic group actions is well defined (and falsifiable), even though it is an interactive assumption. Furthermore, for some groups actions (i.e., ones implied by cryptographic pairings over prime-order groups) the Decisional Diffie-Hellman oracle is publicly computable and hence $\mathsf{GA\text{-}StCDH}$ becomes non-interactive.

(FO) transform [21,23] to an IND-CPA secure PKE scheme (such as GA-EG) to obtain an IND-CCA secure KEM, with provable security in the QROM. The second, non-generic approach is to adapt the well-known (prime-order group) Hashed ElGamal encryption framework of [1] to group actions by "hashing the raw KEM key" to obtain the *Group Action Hashed ElGamal KEM* (GA-HEG). Indeed, [39] proved the security of GA-HEG (called CSIDH-ECIES in [39]) under the GA-StCDH assumption in the ROM.[2] GA-HEG was implicitly and explicitly used in [28,29,39] and its active (IND-CCA) security in the QROM was left as an open problem in [39].[3]

For building an actively secure NIKE, one cannot apply the FO transformation and hence has to resort to adapting the (prime-order group) Hashed Diffie-Hellman NIKE [20] to obtain the *Group Action Hashed Diffie-Hellman* NIKE protocol (GA-HDH). To the best of our knowledge, the active security of the GA-HDH NIKE has not been formally analyzed yet, not even in the ROM. This is in particular unsatisfactory since GA-HDH has already been considered to be used in practical protocols such as Post-Quantum WireGuard [25] and OPTLS [36].

In conclusion, while the IND-CCA security of GA-HEG in the ROM is known to be implied by the GA-StCDH assumption, it remains an open problem to prove its IND-CCA security in the QROM (under any assumption). Similarly, studying the active security of the GA-HDH NIKE in the QROM also remains an open problem.

1.1 Our Contributions

In this paper we study the active security of the Group Action Hashed Diffie-Hellman NIKE GA-HDH and the Group Action Hashed ElGamal KEM GA-HEG in the QROM, and derive variants thereof with improved security guarantees. We now discuss our results in detail. For an overview of our results obtained for KEMs we refer to Figure 1.

GA-HEG KEM AND GA-HDH NIKE. It is easy to see that in the (non-quantum) ROM the active security of GA-HEG is implied by the GA-StCDH assumption. The first main contribution of this paper is to notice that in the QROM one requires a considerably stronger assumptions to prove security of GA-HEG. To this end we define the following two stronger variants of GA-StCDH which differ only in the access to the decision oracle (for implications see Figure 1):

- Partial Quantum access Strong Diffie-Hellman (GA-PQ-StCDH): the first input to the GA-DDH$(\cdot,\cdot)$ oracle is classical and the second is in quantum superposition.
- Full Quantum access Strong Diffie-Hellman (GA-FQ-StCDH): both inputs to the GA-DDH$(\cdot,\cdot)$ oracle are in quantum superposition.

Similar to the QROM, the answer of a quantum superposition query to the two quantum-accessible GA-DDH oracles is also in quantum superposition.

Our first main theorem states that under the GA-FQ-StCDH assumption (full quantum access to the DDH oracle), GA-HEG is IND-CCA secure in the QROM. Furthermore, IND-CCA security in the QROM of GA-HEG implies the GA-PQ-StCDH assumption (partial quantum access to the DDH oracle), hence GA-PQ-StCDH is necessary for GA-HEG's IND-CCA security. The situation for the GA-HDH NIKE is similar, with the difference that "double base" strong assumptions (called GA-DPQ-StCDH and GA-DFQ-StCDH) are required.

This leaves us in the alarming situation that active security of GA-HEG and GA-HDH inherently require a group action CDH assumption with quantum access to the DDH oracle. Due to the quantum access, the latter assumptions cannot be considered as standard assumptions and require further cryptanalysis before we can recommend using GA-HEG KEM and GA-HDH NIKE in practice.

We will now propose two modifications to get security without quantum access to the decision oracles. The first and more efficient modification is using "key confirmation" and only works for KEMs. The second and less efficient modification relies on the "twinning technique" and can be applied to NIKEs and KEMs.

[2] The QROM proof of a variant called CSIDH-PSEC in [39] is flawed (see Appendix E for details).

[3] There also exist IND-CCA secure PKE schemes constructed directly from CSIDH, using additional structure of the elliptic curves. [32] proposed the SimS scheme which is an extension of SiGamal [19] and relies on a non-standard knowledge-of-exponent assumption to achieve IND-CCA security in the standard model. These protocols and assumptions cannot be modeled in the abstract group action framework.

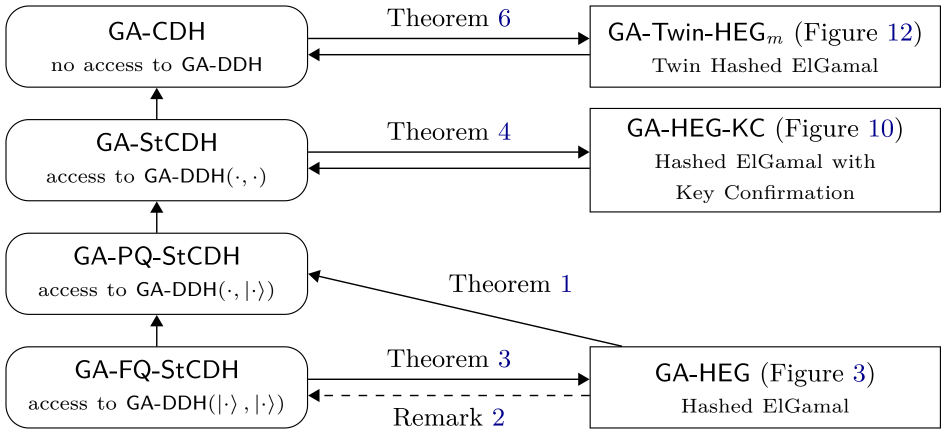

Fig. 1. Overview of our assumptions and results for different variants of hashed ElGamal. The assumptions (elements with rounded corners) are given in Definitions 6 and 7. Solid arrows without indication of a theorem correspond to trivial implications. For the assumptions the only difference is a more limited access to the decision oracle GA-DDH, where $|\cdot\rangle$ denotes quantum access. The dashed arrow holds for *quantum* security, where the adversary is allowed to issue decapsulation queries in superposition.

GA-HEG-KC KEM: Key Confirmation. Our first method is to update GA-HEG the KEM with a key confirmation hash, i.e., every ciphertext additionally contains a hash of the "raw KEM key". This only increases the ciphertext size by one hash, but allows for a different IND-CCA proof technique in the QROM. To be more precise, in the classical ROM, one can use the additional hash to extract the secret information from a ciphertext. In the QROM, this is more involved, but we can use the extractable oracle simulator from [16] to use similar techniques and give a security proof only relying on the more standard GA-StCDH assumption. Specifically, we rely on the fact that decapsulation queries are classical, which allows us to partially measure the simulated random oracle and extract its queries without noticeably disturbing its quantum state.

Unfortunately, it is not possible to use key confirmation in a NIKE setting.

GA-Twin-HEG$_m$ KEM and GA-Twin-HDH$_m$ NIKE: Twinning. We show how to use the twinning technique [11] in the context of group actions to build an actively secure KEM and NIKE from the standard GA-CDH assumption (no DDH oracle access) in the QROM. Since group actions only have limited structure compared to prime-order groups, it seems unavoidable to pursue a bit-wise approach for the twinning technique. Our main leverage is a trapdoor test which allows us to check if several adversarial inputs form a Diffie-Hellman tuple with the challenge elements. The failure probability of this trapdoor test can be reduced to the generic quantum search problem, for which the quantum hardness is optimally bounded by the Grover algorithm. Although this approach does not achieve practical efficiency, it is interesting from a theoretical viewpoint. We specify the twinning parameter m for 128-bit security to instantiate the twinned versions of our GA-Twin-HEG$_m$ KEM and GA-Twin-HDH$_m$ NIKE. At this point we want to highlight that our GA-Twin-HDH$_m$ protocol is the only known NIKE with active security from a standard assumption (without quantum accessible DDH oracles).

Efficiency Comparison. In Table 1 in Section 6, we give an overview of the schemes analyzed in this work and compare them to the FO variant GA-EG-FO of Group Action ElGamal. The KEM variants GA-HEG and GA-Twin-HEG$_m$ share the same minimal ciphertext size but we cannot recommend using them since GA-HEG's security inherently relies on the GA-FQ-StCDH assumption (with quantum accessible DDH oracle) and GA-Twin-HEG$_m$ is computationally very expensive. In comparison, the KEM variants GA-HEG-KC and GA-EG-FO only add one additional hash to the ciphertext but offer security from standard assumptions. Here GA-HEG-KC is preferable since decapsulation is about twice as efficient as in GA-EG-FO (due to FO's re-encryption).

As for the more important case of NIKEs, one either has to use the efficient GA-HDH variant with security under the GA-DFQ-StCDH assumption (with quantum accessible DDH oracles) or use the inefficient GA-Twin-HDH$_m$ NIKE. We leave it as an important open problem to construct a practically efficient actively secure NIKE under a standard hardness assumption.

QROM PROOF DETAILS. One of the standard tools to prove security in the QROM is the O2H [38] lemma, which unfortunately leads to quite loose bounds. Recently, there has been a lot of progress in developing new variants which give tighter bounds, such as the measure-rewind-measure O2H (MRM-O2H) [31] lemma. While these variants give usually tighter bounds, they can often only be applied in more limited scenarios due to additional constraints. In our work we show how to apply MRM to GA-HEG and GA-Twin-HEG$_m$ to obtain tighter bounds than the by applying the original O2H lemma. For proving GA-HEG-KC we need to extract the preimages of the key-confirmation hash. We use the extractable random-oracle simulator of [16], which allows use to to prove it from the GA-StCDH assumption.

For GA-Twin-HEG$_m$ and GA-Twin-HDH$_m$, the main tool to remove the need for the GA-StCDH is the trapdoor test. While it is easy to show its indistinguishability for regular groups in the standard model, it is unclear whether or not a quantum adversary has a significant advantage against the trapdoor test compared to a classical adversary. We solve the second problem by showing that the indistinguishability of the trapdoor test can be (tightly) reduced to the Generic Distinguishing Problem (GDP). This allows us to use well-known results on the hardness of quantum search to bound the advantage of such adversaries and apply the trapdoor test as a substitute for the decision oracle of the GA-StCDH assumption.

1.2 Further Applications

We believe our QROM analysis carries over to the following primitives and constructions.

AUTHENTICATED KEY EXCHANGE. Kawashima et al. as well as de Kock et al. [28,29] translated the Diffie-Hellman based AKE protocol of [13] to the CSIDH setting and proved security in the ROM assuming the GA-StCDH assumption. However, both works left it as an open question to prove security in the QROM. Our analysis demonstrates that this proof will only work assuming (at least partial) quantum access to the decision oracle. In this case, our proof techniques carry over directly. Alternatively, we can also extend the AKE protocol by an additional round to include key confirmation. Using the same technique as in our result on hashed ElGamal with key confirmation will allow to prove security of this extended AKE protocol in the QROM based on the GA-StCDH assumption without quantum access to the decision oracle. However, the additional benefit here is that key confirmation enables explicit authentication, whereas the protocol without key confirmation only achieves implicit authentication.

SIGNCRYPTION AND AUTHENTICATED KEMS. The DH-AKEM which was analyzed in the context of the HPKE standard [3] can easily be translated to the group action setting. The scheme is syntactically a signcryption KEM and will be combined with a symmetric encryption scheme. This construction, also named the authenticated mode of HPKE, was proposed to be used in the Message Layer Security (MLS) secure group messaging protocol [6] and the Encrypted Server Name Indication (ESNI) extension for TLS 1.3 [33]. So far, a post-quantum secure instantiation was not proposed, but our results show how to prove security of a group action based construction in the QROM under GA-FQ-StCDH (full quantum access to the decision oracle). Alternatively, we can also extend the scheme by key confirmation and prove security under GA-StCDH.

POST-QUANTUM SECURE TLS. Currently, there is a great effort in replacing the Diffie-Hellman based approach in the TLS handshake by a post-quantum secure alternative. In order to avoid signature schemes which are rather inefficient, a generic KEM-based approach was considered to allow for an easy instantiation [36], however at the cost of efficiency since it requires an additional round. Instead of signatures, it is also possible to use a NIKE directly, as considered for the case of long-term Diffie-Hellman keys in the OPTLS protocol by Krawczyk and Wee in [30] and in a subsequent IETF draft [34]. In this case, a security analysis of the group-action NIKE in the QROM is crucial and our work provides the first results in this direction, namely that a security proof for group action OPTLS will need to rely at least on the GA-PQ-StCDH assumption (partial quantum access to the decision oracles) and is implied by the GA-FQ-StCDH assumption (full quantum access).

MORE APPLICATIONS. In the group setting, Hashed ElGamal can be used to build multi-recipient multi-message PKE (mmPKE) by using the same randomness for multiple messages. This reduces sender bandwidth and computation substantially and can be used in Continuous Group Key Agreement (CGKA), which underlies modern and scalable Secure Group Messaging (SGM) such as MLS [6] to significantly improve performance [4]. Since GA-HEG has an identical structure, reusing randomness can yield a similar construction with post-quantum security. This is a first step towards efficient, post-quantum secure SGM.

2 Preliminaries

For integers m, n where $m < n$, $[m, n]$ denotes the set $\{m, m+1, ..., n\}$. For $m = 1$, we simply write $[n]$. By $\log(x)$ we denote the logarithm over the reals with base 2. For a (finite) set S, $s \xleftarrow{\$} S$ denotes that s is sampled uniformly and independently at random from S. $y \leftarrow \mathcal{A}(x_1, x_2, ...)$ denotes that on input $x_1, x_2, ...$ the probabilistic algorithm $\mathcal{A}$ returns y. $\mathcal{A}^{\mathrm{O}}$ denotes that algorithm $\mathcal{A}$ has access to oracle O. An adversary is a probabilistic algorithm. We will use code-based games, where $\Pr[\mathrm{G} \Rightarrow 1]$ denotes the probability that the final output of game G is 1. The notation $[\![B]\!]$, where B is a boolean statement, refers to a bit that is 1 if the statement is true and 0 otherwise. For all algorithms and oracles, we implicitly require that they check whether (adversarial) inputs are from the expected input space. If this is not the case, the algorithm (oracle) will simply return a failure symbol $\perp$.

2.1 Key Encapsulation Mechanisms

Syntax. Let $\mathcal{PK}$, $\mathcal{SK}$, $\mathcal{C}$, $\mathcal{K}$ be sets. A *key encapsulation mechanism* $\mathsf{KEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ consists of the following three algorithms

- Gen: The key generation algorithm outputs a public key $\mathsf{pk} \in \mathcal{PK}$ and a secret key $\mathsf{sk} \in \mathcal{SK}$.
- $\mathsf{Encaps}(\mathsf{pk})$: On input a public key pk, the encapsulation algorithm returns a ciphertext $\mathsf{ct} \in \mathcal{C}$ and a key $K \in \mathcal{K}$, where ct is an encapsulation of K.
- $\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})$: On input a secret key sk and a ciphertext ct, the decapsulation algorithm returns a key $K \in \mathcal{K}$ or a special failure symbol $\perp$.

We require perfect correctness, i.e. for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$, $(\mathsf{ct}, K) \leftarrow \mathsf{Encaps}(\mathsf{pk})$, we have $\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct}) = K$.

Definition 1 (Security against Chosen Ciphertext Attacks ($\mathsf{IND\text{-}CCA}$)). *Consider the* $\mathsf{IND\text{-}CCA}$ *security game in Figure 2. For a key encapsulation mechanism* KEM *we define the advantage of* $\mathcal{A}$ *winning the game as*

$$\mathsf{Adv}_{\mathsf{KEM}}^{\mathsf{IND\text{-}CCA}}(\mathcal{A}) := |\Pr[\mathsf{IND\text{-}CCA}(\mathcal{A}) \Rightarrow 1] - 1/2| \,.$$

2.2 Non-Interactive Key Exchange

We recall syntax and the CKS security model of a Non-Interactive Key Exchange (NIKE) scheme, as defined in [11,20].

Syntax. A non-interactive key exchange scheme NIKE consists of three algorithms $\mathsf{NIKE.Setup}$, $\mathsf{NIKE.Gen}$ and $\mathsf{NIKE.SharedKey}$ together with an identity space $\mathcal{ID}$ and a shared key space $\mathcal{SHK}$, where identities in the scheme are only used to track which public key is associated to which user.

- $\mathsf{NIKE.Setup}$: The setup algorithm outputs a set of public parameters pp.
- $\mathsf{NIKE.Gen}(\mathsf{pp}, \mathsf{ID})$: On input pp and $\mathsf{ID} \in \mathcal{ID}$, the key generation algorithm outputs a public key pk and a secret key sk.
- $\mathsf{NIKE.SharedKey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$: On input $\mathsf{ID}_1 \in \mathcal{ID}$ together with a public key pk_1 and $\mathsf{ID}_2 \in \mathcal{ID}$ together with a secret key sk_2, the shared key algorithm outputs a shared key K. In case $\mathsf{ID}_1 = \mathsf{ID}_2$, the algorithm outputs a failure symbol $\perp$.

Game $\mathsf{IND\text{-}CCA}(\mathcal{A})$	**Oracle** $\textsc{Decaps}(\mathsf{ct})$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$	**if** $\mathsf{ct} = \mathsf{ct}^*$
$b \xleftarrow{\$} \{0,1\}$	**return** $\perp$
$(\mathsf{ct}^*, K_0) \leftarrow \mathsf{Encaps}(\mathsf{pk})$	**return** $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$
$K_1 \xleftarrow{\$} \mathcal{K}$	
$b' \leftarrow \mathcal{A}^{\textsc{Decaps}}(\mathsf{pk}, \mathsf{ct}^*, K_b)$	
return $[\![b = b']\!]$	

Fig. 2. The $\mathsf{IND\text{-}CCA}$ game for a key encapsulation mechanism KEM.

Correctness. We require that for any pair of identities $\mathsf{ID}_1, \mathsf{ID}_2 \in \mathcal{ID}$ and any corresponding key pairs $(\mathsf{pk}_1, \mathsf{sk}_1)$ and $(\mathsf{pk}_2, \mathsf{sk}_2)$, it holds that

$$\mathsf{NIKE.SharedKey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2) = \mathsf{NIKE.SharedKey}(\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{ID}_1, \mathsf{sk}_1) \ .$$

CKS Security Model. The security of a NIKE protocol is modeled as a game between a challenger and an adversary $\mathcal{A}$. First, the challenger runs $\mathsf{NIKE.Setup}$ to generate the public parameter pp which it outputs to $\mathcal{A}$. The challenger also draws a random bit b and gives $\mathcal{A}$ access to the following oracles.

- RegisterHonest: $\mathcal{A}$ supplies an identity $\mathsf{ID} \in \mathcal{ID}$ and the challenger runs $\mathsf{NIKE.Gen}(\mathsf{pp}, \mathsf{ID})$ to generate a key pair $(\mathsf{pk}, \mathsf{sk})$. It records $(\mathit{honest}, \mathsf{ID}, \mathsf{pk}, \mathsf{sk})$ and returns the public key pk to $\mathcal{A}$.
- RegisterCorrupt: $\mathcal{A}$ supplies an identity $\mathsf{ID} \in \mathcal{ID}$ and a public key pk and the challenger records $(\mathit{corrupt}, \mathsf{ID}, \mathsf{pk}, \perp)$. If $\mathcal{A}$ issues a query with the same ID again later, only the most recent entry is kept. Note here that we do not require that $\mathcal{A}$ knows the corresponding secret key.
- CorruptReveal: $\mathcal{A}$ supplies two identities ID_1 and ID_2 with the restriction that one identity was registered as *honest* and the other one as *corrupt*, otherwise the oracle returns $\perp$. The challenger looks in its record to fetch the secret key of the honest party and the public key of the corrupted party. If ID_1 was honest, it computes and returns $\mathsf{NIKE.SharedKey}(\mathsf{ID}_2, \mathsf{pk}_2, \mathsf{ID}_1, \mathsf{sk}_1)$ and otherwise $\mathsf{NIKE.SharedKey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$.
- Test: $\mathcal{A}$ supplies two identities ID_1 and ID_2 with the restriction that both were registered as *honest* and $\mathsf{ID}_1 \neq \mathsf{ID}_2$, otherwise the oracle returns $\perp$. The challenger fetches the public key of ID_1 and the secret key of ID_2 from its records and computes $K_0 = \mathsf{NIKE.SharedKey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$. It also chooses a random key $K_1 \xleftarrow{\$} \mathcal{SHK}$ and records it for later. It outputs K_b, depending on the bit b chosen at the beginning. If $b = 1$ and $\mathcal{A}$ queries the same identities again, in either order, the recorded key is output again.

The oracles can be queried adaptively and an arbitrary number of times. We require that no identity that was registered as corrupt can be later registered as honest, and vice versa. Finally, the adversary outputs a bit b'.

Definition 2 (Security of NIKE). *Consider the* CKS *security game as described above. Then the advantage of adversary $\mathcal{A}$ against a non-interactive key exchange scheme* NIKE *is defined as*

$$\mathsf{Adv}^{\mathsf{CKS}}_{\mathsf{NIKE}}(\mathcal{A}) := |\Pr[b = b'] - 1/2| \ .$$

2.3 (Restricted) Effective Group Actions

We recall the definition of (restricted) effective group actions from [2], which provides an abstract framework to build cryptographic primitives relying on isogeny-based assumptions such as CSIDH.

Definition 3 (Group Action). *Let $(\mathcal{G}, \cdot)$ be a group with identity element $e \in \mathcal{G}$, and $\mathcal{X}$ a set. A map*

$$\star : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$$

is a group action if it satisfies the following properties:

1. *Identity: $e \star x = x$ for all $x \in \mathcal{X}$.*
2. *Compatibility: $(g \cdot h) \star x = g \star (h \star x)$ for all $g, h \in \mathcal{G}$ and $x \in \mathcal{X}$.*

Remark 1. Throughout this paper, we only consider group actions, where $\mathcal{G}$ is commutative. Moreover we assume that the group action is regular. This means that for any $x, y \in \mathcal{X}$ there exists precisely one $g \in \mathcal{G}$ satisfying $y = g \star x$.

Definition 4 (Effective Group Action). *Let $(\mathcal{G}, \mathcal{X}, \star)$ be a group action satisfying the following properties:*

1. *$\mathcal{G}$ is finite and there exist efficient (PPT) algorithms for membership testing, equality testing, (random) sampling, group operation and inversion.*
2. *The set $\mathcal{X}$ is finite and there exist efficient algorithms for membership testing and to compute a unique representation.*
3. *There exists a distinguished element $\tilde{x} \in \mathcal{X}$ with known representation.*
4. *There exists an efficient algorithm to evaluate the group action, i.e. to compute $g \star x$ given g and x.*

Then we call $\tilde{x} \in \mathcal{X}$ the origin and $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ an effective group action (EGA)*.*

In practice, the requirements from the definition of EGA are often to strong. Therefore we will consider the weaker notion of restricted effective group actions.

Definition 5 (Restricted Effective Group Action). *Let $(\mathcal{G}, \mathcal{X}, \star)$ be a group action and let $\boldsymbol{g} = (g_1, ..., g_n)$ be a generating set for $\mathcal{G}$. Assume that the following properties are satisfied:*
1. *The group $\mathcal{G}$ is finite and $n = poly(\log(\#\mathcal{G}))$.*
2. *The set $\mathcal{X}$ is finite and there exist efficient algorithms for membership testing and to compute a unique representation.*
3. *There exists a distinguished element $\tilde{x} \in \mathcal{X}$ with known representation.*
4. *There exists an efficient algorithm that given $g_i \in \boldsymbol{g}$ and $x \in \mathcal{X}$, outputs $g_i \star x$ and $g_i^{-1} \star x$.*

Then we call $(\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ a restricted effective group action (REGA)*.*

Alamati et al. [2] introduced the definition of a weak unpredictable group action. We will use a different notation for that property which is syntactically closer to the prime-order group setting. Note that both definitions are equivalent. In particular, we will use the following assumption.

Definition 6 (Group Action Computational Diffie-Hellman Problem). *On input $(g \star \tilde{x}, h \star \tilde{x})$, the group action computational Diffie-Hellman problem* (GA-CDH) *requires to compute the set element $gh \star \tilde{x}$. To an effective group action* EGA*, we associate the advantage function of an adversary $\mathcal{A}$ as*

$$\mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{GA\text{-}CDH}}(\mathcal{A}) := \Pr[\mathcal{A}(g \star \tilde{x}, h \star \tilde{x}) \Rightarrow gh \star \tilde{x}] \ ,$$

where $g, h \xleftarrow{\$} \mathcal{G}$.

The most promising post-quantum secure instantiation of REGAs is provided by CSIDH. We recall its properties in Appendix A.

2.4 QROM Preliminaries

We use different well-known results from post-quantum cryptography. Specifically, our proofs use the oneway-to-hiding [38] (O2H) lemma from [5] and its measure-rewind-measure (MRM) variant from [31] as well as the online extractable quantum random oracle framework from [16]. We recall the MRM O2H lemma below. Further definitions as well as some basic techniques such as random oracle simulation can be found in Appendix B.

Lemma 1 (Measure-Rewind-Measure O2H. Lemma 3.3 in [31]). *Let $\mathsf{G}, \mathsf{H} \colon \mathcal{X} \to \mathcal{Y}$ be random functions, z be a random value, and $\mathcal{S} \subseteq \mathcal{X}$ be a random set such that $\mathsf{G}(x) = \mathsf{H}(x)$ for every $x \notin \mathcal{S}$. The tuple $(\mathsf{G}, \mathsf{H}, \mathcal{S}, z)$ may have arbitrary joint distribution. Furthermore, let $\mathcal{A}^{\mathrm{O}}$ be a unitary/reversible quantum oracle algorithm which queries oracle O with query depth d. Then we can construct an algorithm $\mathsf{Ext}^{\mathsf{G},\mathsf{H}}(z)$ such that the running time of* Ext *is about at most three times the one of $\mathcal{A}^{\mathrm{O}}$ and*

$$\left| \Pr_{\mathsf{H},z}[\mathcal{A}^{\mathsf{H}}(z) \Rightarrow 1] - \Pr_{\mathsf{G},z}[\mathcal{A}^{\mathsf{G}}(z) \Rightarrow 1] \right| \leq 4d \Pr_{\mathsf{G},\mathsf{H},\mathcal{S},z}[\mathcal{S} \cap \mathcal{T} \neq \varnothing \colon \mathcal{T} \leftarrow \mathsf{Ext}^{\mathsf{G},\mathsf{H}}(z)] \,.$$

Some of our proofs rely on the hardness of the Generic Distinguishing Problem (GDP), a decisional variant of the Generic Search Problem (GSP) [40,26,24]. Intuitively, an adversary gets oracle access to a function from some domain $\mathcal{D}$ into $\{0, 1\}$, which is either the all-zero function or a function where the probability that any given point maps to 1 is small (i.e. bounded by some $\lambda \in (0, 1)$), and has to decide which is the case. While the complexity of this problem is clear in the classical case, it is somewhat more difficult in the quantum case. We recall and adapt the well-known bounds to the GDP problem in this section.

Lemma 2 (Generic Distinguishing Problem, decision version of Lemma 2 in [5], Lemma 2.9 from [24]). *Let $\mathsf{F} \colon \mathcal{X} \to \{0, 1\}$ be a random function drawn from a distribution such that $\Pr[\mathsf{F}(x) = 1] \leq \lambda$ for all x and $\mathsf{K} \colon \mathcal{X} \to \{0\}$ be the zero-function. Let $\mathcal{A}$ be a q-query algorithm with query depth d with quantum-access to its oracle. Then*

$$\mathsf{Adv}_{\mathsf{F},q,d}^{\mathsf{GDP}}(\mathcal{A}) := \left| \Pr[\mathsf{GDP}_{\mathsf{F},0}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{GDP}_{\mathsf{F},1}^{\mathcal{A}} \Rightarrow 1] \right| \leq 4\sqrt{(d+1)q\lambda} \,, \tag{1}$$

$\underline{\mathsf{Gen}}$	$\underline{\mathsf{Encaps}(\mathsf{pk})}$	$\underline{\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})}$
$\mathsf{sk} \coloneqq g \xleftarrow{\$} \mathcal{G}$	$r \xleftarrow{\$} \mathcal{G}$	$z \coloneqq \mathsf{sk} \star \mathsf{ct}$
$\mathsf{pk} \coloneqq g \star \tilde{x}$	$\mathsf{ct} \coloneqq r \star \tilde{x}$	$K \coloneqq \mathsf{H}(\mathsf{ct}, z)$
return $(\mathsf{pk}, \mathsf{sk})$	$K \coloneqq \mathsf{H}(\mathsf{ct}, r \star \mathsf{pk})$	**return** K
	return (ct, K)	

Fig. 3. Key encapsulation mechanism $\mathsf{GA\text{-}HEG}$ for an effective group action $\mathsf{EGA} = (\mathcal{G}, \mathcal{X}, \star, \tilde{x})$, where $\mathsf{H} : \mathcal{X} \times \mathcal{X} \to \{0,1\}^\kappa$ is a hash function.

where $\mathsf{GDP}^{\mathcal{A}}_{\mathsf{F},0} \coloneqq \mathcal{A}^{\mathsf{K}}()$ *and* $\mathsf{GDP}^{\mathcal{A}}_{\mathsf{F},1} \coloneqq \mathcal{A}^{\mathsf{F}}()$. *Moreover, if the outputs of* F *are independent we have*

$$\mathsf{Adv}^{\mathsf{GDP}}_{\mathsf{F},q,d}(\mathcal{A}) \leq 8(q+1)^2 \lambda\,. \tag{2}$$

We prove eq. (1) in Appendix B.2. The bound in eq. (2) is a reformulation from Lemma 2.9 from [24].

3 Necessary Assumptions for Group Action KEM and NIKE in the QROM

In this section we will first recall the two schemes we are looking at: Group Action Hashed ElGamal and the Group Action Hashed Diffie-Hellman NIKE scheme. We denote the schemes by $\mathsf{GA\text{-}HEG}$ and $\mathsf{GA\text{-}HDH}$, respectively.

Group Action Hashed ElGamal. The scheme is given in Figure 3. Note that this is the same scheme as the CSIDH-ECIES-KEM considered in [39]. The public parameters consist of an effective group action $\mathsf{EGA} = (\mathcal{G}, \mathcal{X}, \star, \tilde{x})$ and a hash function $\mathsf{H} : \mathcal{X}^2 \to \{0,1\}^\kappa$. Further we set $\mathcal{PK} = \mathcal{X}$, $\mathcal{SK} = \mathcal{G}$ and $\mathcal{K} = \{0,1\}^\kappa$. The key generation algorithm samples a random group element $g \xleftarrow{\$} \mathcal{G}$ as secret key. In order to compute the public key, g is applied to the origin element $\tilde{x}$ using the group action operation. The set element $\mathsf{pk} = g \star \tilde{x}$ is the public key. The encapsulation algorithm also first samples a random group element $r \xleftarrow{\$} \mathcal{G}$ and then calculates the ciphertext $\mathsf{ct} = r \star \tilde{x}$. The key is derived by first computing $r \star \mathsf{pk}$ (the shared DH value) and subsequently hashing $r \star \mathsf{pk}$ together with the ciphertext ct. Decapsulation first recomputes the shared DH value $g \star \mathsf{ct} = r \star \mathsf{pk}$ and then applies the hash function H. Correctness of the scheme holds due to the commutativity of the group action.

Group Action Hashed Diffie-Hellman. A schematic overview of the hashed Diffie-Hellman NIKE scheme $\mathsf{GA\text{-}HDH}$ is given in Figure 4. As in the hashed ElGamal scheme, the public parameters pp include the description of EGA together with a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\kappa$ such that $\mathcal{PK} = \mathcal{X}$, $\mathcal{SK} = \mathcal{G}$ and $\mathcal{SHK} = \{0,1\}^\kappa$. We assume that $\mathcal{ID} = \{0,1\}^\mu$, which means that each identity is represented by a bitstring of length μ and there is a natural ordering $<$ on the space of identities. On input an $\mathsf{ID} \in \mathcal{ID}$, the key generation algorithm chooses a group element $g \xleftarrow{\$} \mathcal{G}$ which will be the secret key $\mathsf{sk}_{\mathsf{ID}}$. The public key is computed as $\mathsf{pk}_{\mathsf{ID}} = g \star \tilde{x} \in \mathcal{X}$. The shared key of an identity ID_1 with public key $\mathsf{pk}_{\mathsf{ID}_1} = x$ and an identity $\mathsf{ID}_2 \neq \mathsf{ID}_1$ with secret key $\mathsf{sk}_{\mathsf{ID}_2} = g$ is defined as

$$K = \begin{cases} \mathsf{H}(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_{\mathsf{ID}_1}, \mathsf{pk}_{\mathsf{ID}_2}, g \star x) & \textbf{if } \mathsf{ID}_1 < \mathsf{ID}_2 \\ \mathsf{H}(\mathsf{ID}_2, \mathsf{ID}_1, \mathsf{pk}_{\mathsf{ID}_2}, \mathsf{pk}_{\mathsf{ID}_1}, g \star x) & \textbf{if } \mathsf{ID}_2 < \mathsf{ID}_1 \end{cases}.$$

Correctness again holds because of the commutativity of the group action itself and the ordering of IDs.

One of the goals of this work is to prove these schemes secure in the QROM (cf. Section 4). However, as it turns out, we will need stronger assumptions for the proofs than those defined in the literature. In the next section we introduce the corresponding assumptions. Furthermore, we show that a (somewhat) stronger assumption is indeed necessary by showing that it is implied by the security of the schemes themselves.

3.1 Computational Group Action Diffie-Hellman with Quantum Oracle Access

Our new assumptions are all variants of the group action strong computational Diffie-Hellman problem ($\mathsf{GA\text{-}StCDH}$). The $\mathsf{GA\text{-}StCDH}$ assumption is basically the translation of the strong CDH problem to

Alice A	**Bob** B
$\mathsf{sk_A} = a \xleftarrow{\$} \mathcal{G}$	$\mathsf{sk_B} = b \xleftarrow{\$} \mathcal{G}$
$\mathsf{pk_A} = a \star \tilde{x}$	$\mathsf{pk_B} = b \star \tilde{x}$
$z \coloneqq a \star \mathsf{pk_B}$	$z \coloneqq b \star \mathsf{pk_A}$
$K \coloneqq \mathsf{H}(\mathsf{A}, \mathsf{B}, \mathsf{pk_A}, \mathsf{pk_B}, z)$	

Fig. 4. Group Action Non-Interactive Key Exchange scheme GA-HDH for an effective group action $\mathsf{EGA} = (\mathcal{G}, \mathcal{X}, \star, \tilde{x})$, where $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\kappa$ is a hash function.

group actions (cf. also [28,29]), where the adversary is given access to a (fixed-base) decision oracle. What we need for our proofs is actually *quantum* access to the decision oracle, which is a considerably stronger assumption that was never considered before. For the NIKE proofs, we will also need a double-sided oracle definition, where the adversary gets access to two decision oracles, one for each of the challenge set elements, and its quantum variants. All variants are captured by Definition 7.

Definition 7 (Variants of GA-StCDH). *On input $(g\star\tilde{x}, h\star\tilde{x})$, the* GA-XXX-StCDH *requires to compute the set element $gh\star\tilde{x}$ with access to a decision oracle which is specified below. To an effective group action* EGA *and an adversary $\mathcal{A}$, we associate the advantage function*

$$\mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{GA\text{-}XXX\text{-}StCDH}}(\mathcal{A}) \coloneqq \Pr[\mathcal{A}^{\mathrm{O}}(g \star \tilde{x}, h \star \tilde{x}) \Rightarrow gh \star \tilde{x}] \ ,$$

where $g, h \xleftarrow{\$} \mathcal{G}$ and

$$\mathrm{O} \coloneqq \begin{cases} \mathsf{GA\text{-}DDH}_g(\cdot,\cdot)\,, & \mathsf{XXX} = \{\} & \textit{(classical)} \\ \mathsf{GA\text{-}DDH}_g(\cdot,|\cdot\rangle)\,, & \mathsf{XXX} = \mathsf{PQ} & \textit{(partially quantum)} \\ \mathsf{GA\text{-}DDH}_g(|\cdot\rangle\,,|\cdot\rangle)\,, & \mathsf{XXX} = \mathsf{FQ} & \textit{(fully quantum)} \\ \{\mathsf{GA\text{-}DDH}_g(\cdot,\cdot), \mathsf{GA\text{-}DDH}_h(\cdot,\cdot)\}\,, & \mathsf{XXX} = \mathsf{D} & \textit{(double-sided classical)} \\ \{\mathsf{GA\text{-}DDH}_g(\cdot,|\cdot\rangle), \mathsf{GA\text{-}DDH}_h(\cdot,|\cdot\rangle)\}\,, & \mathsf{XXX} = \mathsf{DPQ} & \textit{(double-sided partially quantum)} \\ \{\mathsf{GA\text{-}DDH}_g(|\cdot\rangle\,,|\cdot\rangle), \mathsf{GA\text{-}DDH}_h(|\cdot\rangle\,,|\cdot\rangle)\}\,, & \mathsf{XXX} = \mathsf{DFQ} & \textit{(double-sided fully quantum)} \end{cases}$$

On basis-state inputs (y, z), $\mathsf{GA\text{-}DDH}_g$ *returns 1 if $g \star y = z$ and 0 otherwise.* $\mathsf{GA\text{-}DDH}_h$ *is defined equivalently. Note that superposition queries are implicitly then defined by linearity (i.e., $\mathrm{O}(\sum_x \alpha_x x) = \sum_x \alpha_x \mathrm{O}(x)$). We emphasize that the partially quantum variants of the oracle measure their corresponding first input implicitly.*

3.2 Necessity of the GA-(D)PQ-StCDH Assumption

We now show that partial quantum access to the decision oracle is indeed a *necessary* assumption to prove IND-CCA security of GA-HEG and CKS security of GA-HDH. We do that by showing the opposite direction, namely that the assumption is implied by the security of the corresponding scheme. This is captured by the following two theorems.

Theorem 1. *Let $\mathsf{H}\colon \mathcal{X} \times \mathcal{X} \to \{0,1\}^\kappa$ be a random oracle. For any quantum adversary $\mathcal{A}$ against* GA-PQ-StCDH *making at most q queries to its decision oracle, there exists a quantum adversary $\mathcal{B}$ against* IND-CCA *security of* GA-HEG *making at most q decapsulation queries and $q+1$ quantum random oracle queries with*

$$\mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{GA\text{-}PQ\text{-}StCDH}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}_{\mathsf{GA\text{-}HEG}}^{\mathsf{IND\text{-}CCA}}(\mathcal{B}) + \frac{8(q+1)^2+1}{2^\kappa}\,,$$

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

Theorem 2. *Let $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^\kappa$ be a random oracle. For any quantum adversary $\mathcal{A}$ against* GA-DPQ-StCDH *making at most q queries to its decision oracles, there exists a quantum adversary $\mathcal{B}$ against the* CKS *security of* GA-HDH *making 2 queries to the* RegisterHonest *oracle, at most q queries to the* RegisterCorrupt *oracle and $q+1$ quantum random oracle queries with*

$$\mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{GA\text{-}DPQ\text{-}StCDH}}(\mathcal{A}) \leq 2 \cdot \mathsf{Adv}_{\mathsf{GA\text{-}HDH}}^{\mathsf{CKS}}(\mathcal{B}) + \frac{8(q+1)^2+1}{2^\kappa}\,,$$

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

Games G_1-G_5		**Oracle** $\mathrm{O}(x_1, x_2)$	
00 $g \xleftarrow{\$} \mathcal{G}$		05 Let $a := e$	$\backslash\backslash\, \mathrm{G}_2$-$\mathrm{G}_5$
01 $h \xleftarrow{\$} \mathcal{G}$		06 **if** $x_1 = h \star \tilde{x}$: Let $a := \hat{g}$	$\backslash\backslash\, \mathrm{G}_3$-$\mathrm{G}_5$
02 $\hat{g} \xleftarrow{\$} \mathcal{G} \setminus \{e\}$	$\backslash\backslash\, \mathrm{G}_3$-$\mathrm{G}_5$	07 **return** $[\![\mathsf{Decaps}(g, a \star x_1) = \mathsf{H}(a \star x_1, a \star x_2)]\!]$	$\backslash\backslash\, \mathrm{G}_5$
03 $z \leftarrow \mathcal{A}^{\mathrm{O}(\cdot,\lvert\cdot\rangle)}(g \star \tilde{x}, h \star \tilde{x})$		08 **return** $[\![\mathsf{H}(a \star x_1, (a \cdot g) \star x_1) = \mathsf{H}(a \star x_1, a \star x_2)]\!]$	$\backslash\backslash\, \mathrm{G}_4$
04 **return** $[\![z = gh \star \tilde{x}]\!]$		09 **return** $[\![(a \star x_1, (a \cdot g) \star x_1) = (a \star x_1, a \star x_2)]\!]$	$\backslash\backslash\, \mathrm{G}_2$-$\mathrm{G}_3$
		10 **return** $[\![g \star x_1 = x_2]\!]$	$\backslash\backslash\, \mathrm{G}_1$

Fig. 5. Games G_1-G_5 for the proof of Theorem 1.

We will prove Theorem 1 below. The proof of Theorem 2 is very similar and we refer to Appendix D.2 for more details.

Proof (of Theorem 1). The idea of the proof is to construct a reduction which implements the decision oracle using the decapsulation oracle by testing whether $\mathsf{Decaps}(x_1) = \mathsf{H}(x_1, x_2)$ on a decision oracle query $\mathrm{O}(x_1, x_2)$. Whenever $\mathrm{O}(x_1, x_2)$ returns 1, so will $\mathsf{Decaps}(x_1) = \mathsf{H}(x_1, x_2)$, except when x_1 is the challenge ciphertext. Therefore, whenever x_1 is the challenge ciphertext, the reduction is going to do the same test, except that it first "shifts" x_1 and x_2 by some other group element $\hat{g}$. After simulating all decision oracle queries, the reduction returns whether the challenge KEM key K does not equal $\mathsf{H}(c^*, z)$ where z is the group action CDH solution obtained by $\mathcal{A}$. We now proceed with the formal proof.

Let $\mathcal{A}$ be a quantum adversary as described in Theorem 1. Consider the sequence of games given in Figure 5.

Game G_1. This is the $\mathsf{GA\text{-}PQ\text{-}StCDH}$ game, where $\mathrm{O} = \mathsf{GA\text{-}DDH}_g$. By definition,

$$\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] = \mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{GA\text{-}PQ\text{-}StCDH}}(\mathcal{A}) \,.$$

Game G_2. In this game, instead of returning whether $g \star x_1 = x_2$, the decision oracle returns whether $(x_1, g \star x_1) = (x_1, x_2)$. In order to prepare for the next game hop, we additionally introduce a new variable a which denotes a group element. In G_2, a is always the neutral element e of $\mathcal{G}$, thus applying a on any set element does not have any effect. Since we always have $x_1 = x_1$, the check in line 09 is the same as in line 10. Hence we have $\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1]$.

Game G_3. In this game we sample a group element $\hat{g} \xleftarrow{\$} \mathcal{G} \setminus \{e\}$ uniformly at random in line 02. For all queries (x_1, x_2) to O, where $x_1 = h \star \tilde{x}$, we now set a to $\hat{g}$. In this case, this will change the boolean test in line 09. However, since the group action operation is a bijection, this change is only conceptual. The reason for doing this, is that in the final reduction we are going to set $h \star \tilde{x}$ to be the challenge ciphertext c^* which we cannot query to the decapsulation oracle. Shifting by $\hat{g}$ in the case that $x_1 = h \star \tilde{x}$ will allow us to still simulate O. We get $\Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1]$.

Game G_4. In this game we perform the boolean test by first hashing both sides using a random oracle. In particular, we check if $\mathsf{H}(a \star x_1, (a \star g) \star x_1) = \mathsf{H}(a \star x_1, a \star x_2)$ in line 08. This introduces false positives into the decision oracle, when for any $\hat{x}_1 \in \mathcal{X}$ we have that $\mathsf{H}(\hat{x}_1, g \star \hat{x}_1)$ has preimages of the form $(\hat{x}_1, \hat{x}_2)$ with $\hat{x}_2 \neq g \star \hat{x}_1$. We can bound this change by reducing to the GDP problem, which we do in Figure 6. In particular, for every $(\hat{x}_1, \hat{x}_2)$ we have $\mathsf{F}(\hat{x}_1, \hat{x}_2)$ returns 1 with probability $\lambda := 1/2^{\kappa}$, which is the probability to find a second preimage for $\mathsf{H}(\hat{x}_1, g \star \hat{x}_1)$. If F is the zero function, the distinguisher $\mathcal{D}$ simulates G_3 and otherwise it simulates G_4. Thus by eq. (2) of Lemma 2 where we have set $\lambda := 1/2^{\kappa}$ we have

$$\begin{aligned} &\left|\Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1]\right| \\ = &\left|\Pr[\mathsf{GDP}_{\mathsf{F},0}^{\mathcal{D}} \Rightarrow 1] - \Pr[\mathsf{GDP}_{\mathsf{F},1}^{\mathcal{D}} \Rightarrow 1]\right| \leq 8(q+1)^2/2^{\kappa} \,. \end{aligned}$$

Game G_5. In this game we change the boolean test again and check whether $\mathsf{Decaps}(g, a \star x_1) = \mathsf{H}(a \star x_1, a \star x_2)$ in line 07. By definition of decapsulation, this change is again only conceptual. We have $\Pr[\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_5^{\mathcal{A}} \Rightarrow 1]$.

It remains to bound G_5. We claim

$$\Pr[\mathrm{G}_5^{\mathcal{A}} \Rightarrow 1] \leq 2 \cdot \mathsf{Adv}_{\mathsf{GA\text{-}HEG}}^{\mathsf{IND\text{-}CCA}}(\mathcal{B}) + 1/2^{\kappa} \,. \tag{3}$$

Distinguisher $\mathcal{D}^{\mathsf{F}}$	**Oracle** $\mathrm{O}(x_1, x_2)$
$g \xleftarrow{\$} \mathcal{G}$	**if** $g \star x_1 = x_2$ **return** 1
$h \xleftarrow{\$} \mathcal{G}$	**if** $x_1 = h \star \tilde{x}$
$\hat{g} \xleftarrow{\$} \mathcal{G} \setminus \{e\}$	**return** $\mathsf{F}(\hat{g} \star x_1, \hat{g} \star x_2)$
$z \leftarrow \mathcal{A}^{\mathrm{O}(\cdot, \|\cdot\rangle)}(g \star \tilde{x}, h \star \tilde{x})$	**else**
return $[\![z = gh \star \tilde{x}]\!]$	**return** $\mathsf{F}(x_1, x_2)$

Fig. 6. Distinguisher $\mathcal{D}$ for the Generic Distinguishing Problem to bound G_4-G_5.

Adversary $\mathcal{B}^{\textsc{Decaps},\mathsf{H}}(\mathsf{pk}, c^*, K)$	**Oracle** $\mathrm{O}(x_1, x_2)$
$\hat{g} \xleftarrow{\$} \mathcal{G} \setminus \{e\}$	**if** $x_1 = c^*$
$z \leftarrow \mathcal{A}^{\mathrm{O}(\cdot, \|\cdot\rangle)}(\mathsf{pk}, c^*)$	**return** $[\![\textsc{Decaps}(\hat{g} \star x_1) = \mathsf{H}(\hat{g} \star x_1, \hat{g} \star x_2)]\!]$
return $[\![K \neq \mathsf{H}(c^*, z)]\!]$	**return** $[\![\textsc{Decaps}(x_1) = \mathsf{H}(x_1, x_2)]\!]$

Fig. 7. Adversary $\mathcal{B}$ against IND-CCA security for bounding G_6.

The adversary $\mathcal{B}$ in Figure 7 simulates G_5 as follows: it runs $\mathcal{A}$ on its own inputs (pk, c^*), thus defining $g \star \tilde{x} := \mathsf{pk}$ and $h \star \tilde{x} := c^*$. Note that it can simulate oracle O as in G_5 using its own Decaps oracle and random oracle H provided by the IND-CCA challenger. If $\mathcal{A}$ queries O on the challenge ciphertext c^*, we make use of the additional element $\hat{g}$, thus $\mathcal{B}$ never queries Decaps on the challenge ciphertext. Finally $\mathcal{A}$ outputs z. If $\mathsf{H}(c^*, z) = K^*$, where K^* is the challenge key $\mathcal{B}$ received at the beginning, it returns 0 (real), otherwise it returns $b' := 1$ (random). Clearly, if $\mathcal{A}$ computes z as $gh \star \tilde{x}$, $\mathcal{B}$ always wins the IND-CCA game when it is in the real world. In the random world, it will win only with probability $1 - 1/2^{\kappa}$ since the challenge key might be the same as the real key with probability $1/2^{\kappa}$. When z is not the correct solution and K is the real key, then $\mathcal{B}$ will only win if the output of H still coincides with K, i.e. with probability $1/2^{\kappa}$. However, if K is a random key, $\mathcal{B}$ will win again with probability $1 - 1/2^{\kappa}$. Collecting the conditional probabilities yields the bound claimed in eq. (3).

It remains to analyze the running time of $\mathcal{B}$ and its additional oracle calls. $\mathcal{B}$ runs $\mathcal{A}$ once and for every query to O, $\mathcal{B}$ makes one call to the decapsulation oracle and random oracle. After running $\mathcal{A}$ it makes one additional call to the random oracle, which yields the claimed number of additional oracle calls, which concludes our proof. □

Remark 2. Quantum-secure signatures and public-key encryption schemes have been studied in [10], where the adversary gets quantum access to the signing and decryption oracle, respectively. One can show that the *Quantum* IND-CCA (IND-qCCA) security of GA-HEG is equivalent to the GA-FQ-StCDH assumption, that is the assumption is necessary and sufficient. The proof that IND-qCCA implies the GA-FQ-StCDH assumption is the same as the proof of Theorem 1. Therefore, observe that since the first input of the decision oracle is not measured, the reduction needs a quantum-accessible decapsulation oracle, which is provided by the IND-qCCA game. The sufficiency follows by observing that the reduction in the proof of Theorem 3 can actually simulate quantum decapsulation queries. We leave it as an open problem whether the GA-PQ-StCDH assumption is sufficient for IND-CCA security GA-HEG.

4 Security of Group Action Hashed ElGamal and NIKE

We now prove security of the two schemes in the quantum random oracle model. In particular, we prove IND-CCA security of GA-HEG under the GA-FQ-StCDH assumption and CKS security of GA-HDH under the GA-DFQ-StCDH assumption, i.e., with full quantum access to the decision oracle.

Due to our results in Section 3.2, we cannot hope to prove security of the (un-modified) schemes based on assumptions without quantum access. However, adding key confirmation to GA-HEG allows us to do so. We elaborate in more detail in Section 4.2. Unfortunately, key confirmation cannot be applied in the context of non-interactive schemes such as GA-HDH.

4.1 Security of GA-HEG

The following theorem states security of GA-HEG based on the GA-FQ-StCDH assumption. For the proof we will use the MRM O2H lemma (Lemma 1).

Games G_1-G_5

00 $\mathsf{sk} := g \xleftarrow{\$} \mathcal{G}$
01 $\mathsf{pk} := x := g \star \tilde{x}$
02 $b \xleftarrow{\$} \{0,1\}$
03 $r \xleftarrow{\$} \mathcal{G}$
04 $c^* := r \star \tilde{x}$
05 $K_0 := \mathsf{H}(c^*, r \star \mathsf{pk})$
06 $\mathsf{H}[(c^*, r \star \mathsf{pk})] \xleftarrow{\$} \{0,1\}^\kappa$ $\quad\backslash\!\backslash\, \mathrm{G}_5$
07 $K_1 \xleftarrow{\$} \{0,1\}^\kappa$
08 $b' \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{Dec}}(\mathsf{pk}, c^*, K_b)$
09 **return** $[\![b = b']\!]$

Oracle $\mathsf{Decaps}(\mathsf{sk}, c)$

10 **if** $c = c^*$ **return** $\perp$
11 **return** $\mathsf{H}_1(c)$ $\quad\backslash\!\backslash\, \mathrm{G}_4$-$\mathrm{G}_5$
12 **return** $\mathsf{H}(c, \mathsf{sk} \star c)$

Oracle $\mathsf{H}(x_1, x_2)$ $\quad\backslash\!\backslash\, \mathrm{G}_2$-$\mathrm{G}_5$

13 **if** $(x_1, x_2) = (x_1, g \star x_1)$
14 $\quad$ **return** $\mathsf{H}_1(x_1)$ $\quad\backslash\!\backslash\, \mathrm{G}_3$-$\mathrm{G}_5$
15 $\quad$ **return** $\mathsf{H}_1(x_1, x_2)$
16 **return** $\mathsf{H}_2(x_1, x_2)$

Fig. 8. Games G_1-G_5 for the proof of Theorem 3, where H_1 and H_2 are internal random oracles.

Remark 3. Alternatively, we could use the O2H variant of [8] (also for proving $\mathsf{GA\text{-}Twin\text{-}HEG}_m$) by using its extractor in the proof, yielding a bound of $\sqrt{\mathsf{Adv}}$. Since both versions are applicable, one can essentially choose between a quadratic loss independent of the adversary's query depth or a linear loss in the query depth. To keep proofs and theorems simple, we only prove the bound using MRM.

Theorem 3. *For any quantum adversary $\mathcal{A}$ against* $\mathsf{IND\text{-}CCA}$ *security of* $\mathsf{GA\text{-}HEG}$ *that issues at most q queries to the quantum-accessible random oracle* H *of query depth d with query parallelism $p := q/d$, there exists an adversary $\mathcal{B}$ against* $\mathsf{GA\text{-}FQ\text{-}StCDH}$ *such that*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{GA\text{-}HEG}}(\mathcal{A}) \leq 4d\mathsf{Adv}^{\mathsf{GA\text{-}FQ\text{-}StCDH}}_{\mathsf{EGA}}(\mathcal{B}),$$

and the running time of $\mathcal{B}$ is about three times that of $\mathcal{A}$ plus at most $\mathcal{O}(q + p)$ queries to the decision oracle and the time to simulate up to $\mathcal{O}(\max\{q_D, q\})$ random oracle queries, where q_D is the number of decapsulation queries.

Proof. Let $\mathcal{A}$ be a quantum adversary as described in Theorem 3. Consider the games given in Figure 8. We proceed by analyzing the different games.

Game G_1. This is the $\mathsf{IND\text{-}CCA}$ game where we unfolded the definition of $\mathsf{GA\text{-}HEG}$. By definition,

$$\left|\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] - 1/2\right| = \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{GA\text{-}HEG}}(\mathcal{A}).$$

Game G_2. Here we introduce the following conceptual change: the random oracle H is simulated using two *internal* random oracles H_1 and H_2, where the first one is used on valid DH tuples, and the second on invalid ones. For this change to be meaningful (i.e., simulatable) later on, we need a quantum-accessible decision oracle, which is provided by the $\mathsf{GA\text{-}FQ\text{-}StCDH}$ assumption. Clearly, the change is only conceptual and we have $\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1]$.

Game G_3. Next, we drop the input x_2 in the case where the random oracle H_1 is used, that is we return $\mathsf{H}_1(x_1)$ instead of $\mathsf{H}_1(x_1, x_2)$. Since relative to pk and x_1 there exists a *unique* x_2 s.t. $(x_1, x_2) = (x_1, g \star x_1)$, due to the regularity property of EGA, this change is again only conceptual and we have $\Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1]$.

Game G_4. In this game we remove the usage of the secret key in the random oracle calls of the decapsulation oracle by returning $\mathsf{H}_1(c)$ instead of $\mathsf{H}(c, g \star c)$. Note that the secret key is only used to check for the DDH condition, which can be simulated with access to $\mathsf{GA\text{-}DDH}_g(|\cdot\rangle, |\cdot\rangle)$. Due to the previous conceptual change $\mathsf{H}_1(c) = \mathsf{H}(c, g \star c)$ holds by definition and therefore this change is again only conceptual, thus $\Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1]$.

Game G_5. In this game we reprogram the random oracle on the challenge input $(c^*, r \star \mathsf{pk})$, after querying $\mathsf{H}(c^*, r \star \mathsf{pk})$ in line 06. Now K_0 is identicially distributed as K_1, therefore the key is now independent of the challenge bit b and we have $\Pr[\mathrm{G}_5^{\mathcal{A}} \Rightarrow 1] = 1/2$. Due to Lemma 1 (MRM-O2H) we have

$$\left|\Pr[\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_5^{\mathcal{A}} \Rightarrow 1]\right| \leq 4d \Pr[\mathrm{G}_6^{\mathsf{Ext}} \Rightarrow 1],$$

where $\mathrm{G}_6^{\mathsf{Ext}}$ is like $\mathrm{G}_4^{\mathcal{A}}$, except that instead of running $\mathcal{A}$, it runs the extraction algorithm $\mathsf{Ext}^{\mathsf{Decaps},\mathsf{H},\mathsf{H}'}$ from the MRM-O2H lemma to obtain a set $\mathcal{T}$ and the winning condition is changed to $[\![\mathcal{S} \cap \mathcal{T} \neq \varnothing]\!]$, where $\mathcal{S} := \{(c^*, r \star \mathsf{pk})\}$ and H' is the reprogrammed random oracle.

Adversary $\mathcal{B}^{\lvert\mathrm{O}\rangle}(g\star\tilde{x}, h\star\tilde{x})$	**Oracle** $\mathsf{Decaps}(\mathsf{sk}, c)$
$\mathsf{pk} := g\star\tilde{x},\ c^* := h\star\tilde{x}$	**if** $c = c^*$ **return** $\bot$
$K_0, K_1 \xleftarrow{\$} \{0,1\}^\kappa,\ b \xleftarrow{\$} \{0,1\}$	**return** $\mathsf{H}_1(c)$
$\mathcal{T} \leftarrow \mathsf{Ext}^{\mathsf{H},\mathsf{H}',\mathsf{Dec}}(\mathsf{pk}, c^*, K_b)$	**Oracle** $\mathsf{H}/\mathsf{H}'(x_1, x_2)$
for $(a, z) \in \mathcal{T}$ $\quad\backslash\backslash\ \lvert\mathcal{T}\rvert = p$	**if** $\mathrm{O}(x_1, x_2) = 1$
$\quad$ **if** $a = h\star\tilde{x} \wedge \mathrm{O}(a, z) = 1$	$\quad$ **if** $x_1 = c^*$ **return** K_0 $\quad\backslash\backslash\ \mathsf{H}$ only
$\quad\quad$ **return** z $\quad\backslash\backslash = gh\star\tilde{x}$	$\quad$ **return** $\mathsf{H}_1(x_1)$
return $\bot$	**return** $\mathsf{H}_2(x_1, x_2)$

Fig. 9. Adversary $\mathcal{B}$ for the game-hop G_4-G_5 for the proof of Theorem 3. H_1 and H_2 are internal random oracles. The oracle O is the $\mathsf{GA\text{-}DDH}_g$ oracle.

$\underline{\mathsf{Gen}}$	$\underline{\mathsf{Encaps}(\mathsf{pk})}$	$\underline{\mathsf{Decaps}(\mathsf{sk}, \mathsf{ct})}$
$\mathsf{sk} := g \xleftarrow{\$} \mathcal{G}$	$r \xleftarrow{\$} \mathcal{G}$	$z := \mathsf{sk}\star c$
$\mathsf{pk} := x := g\star\tilde{x}$	$c := r\star\tilde{x}$	**if** $\mathsf{G}(c, z) \neq d$
return $(\mathsf{pk}, \mathsf{sk})$	$d := \mathsf{G}(c, r\star\mathsf{pk})$	$\quad$ **return** $\bot$
	$K := \mathsf{H}(c, r\star\mathsf{pk})$	$K := \mathsf{H}(c, z)$
	return $(\mathsf{ct} := (c, d), K)$	**return** K

Fig. 10. Key encapsulation mechanism $\mathsf{GA\text{-}HEG\text{-}KC}$ for an effective group action $\mathsf{EGA} = (\mathcal{G}, \mathcal{X}, \star, \tilde{x})$, where $\mathsf{G} : \mathcal{X}\times\mathcal{X} \to \{0,1\}^n$ and $\mathsf{H} : \mathcal{X}\times\mathcal{X} \to \{0,1\}^\kappa$ are hash functions.

We bound the right-hand probability by the adversary $\mathcal{B}$ given in Figure 9, which runs the extraction algorithm simulating Decaps and H as in G_4 and H' (the reprogrammed H) as in G_5. Observe that $\mathcal{B}$ can simulate quantum decapsulation queries, since it has quantum access to H_1, which is why we can apply the MRM-O2H lemma. Since $\mathcal{B}$ wins if $\mathcal{S}\cap\mathcal{T} \neq \varnothing$, we have

$$\Pr[\mathrm{G}_6^{\mathsf{Ext}} \Rightarrow 1] \leq \mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{GA\text{-}FQ\text{-}StCDH}}(\mathcal{B})\,.$$

Combining all inequalities yields the claimed bound. We conclude our proof by analyzing the running time of $\mathcal{B}$. $\mathcal{B}$ runs the extraction algorithm Ext, whose running time is at most three times that of $\mathcal{A}$. For every run of $\mathcal{A}$, it has to simulate at most $\max\{q_D, q\}$ calls to H_1 and q calls to H_2 (through H, H'), where it calls O on every query. Then, after obtaining $\mathcal{T}$, it makes at most p queries to O, thus $q + p$ total queries to O. Multiplying the parts of simulating $\mathcal{A}$ by 3, adding up and applying $\mathcal{O}$ notation yields the claimed running time and additional oracle calls, which concludes our proof. $\square$

4.2 Security of GA-HEG via Key Confirmation

We recall the Hashed ElGamal scheme with key confirmation in Figure 10. We denote this scheme by $\mathsf{GA\text{-}HEG\text{-}KC}$. Compared to the original scheme in Figure 3, we now have a second hash function $\mathsf{G}\colon \mathcal{X}\times\mathcal{X} \to \{0,1\}^n$ which is used to compute an additional ciphertext element d. The input to this hash function is the same as for the final key. The decapsulation algorithm now first checks if d is valid by recomputing it. If this check passes, the actual key is computed and returned, otherwise the algorithm outputs a failure symbol $\bot$.

Theorem 4 establishes security of $\mathsf{GA\text{-}HEG\text{-}KC}$ based on the $\mathsf{GA\text{-}StCDH}$ assumption, that is without quantum access to the decision oracle. One reason for the looser bound is that the classical decision oracle does not enable us to apply the more recent O2H lemmata. The other is that we have to first apply O2H, before applying the extractable RO simulator.

Theorem 4. *Let* $\mathsf{G}\colon \mathcal{X}\times\mathcal{X} \to \{0,1\}^n$ *be a random oracle. For any quantum adversary* $\mathcal{A}$ *against* $\mathsf{IND\text{-}CCA}$ *security of* $\mathsf{GA\text{-}HEG\text{-}KC}$ *that issues at most d parallel queries each of size p (in total $q := dp$ queries) to the quantum-accessible random oracles* H *and* G *and* q_D *decapsulation queries, there exists an* *adversary $\mathcal{B}$ against the* GA-StCDH *such that*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{GA\text{-}HEG\text{-}KC}}(\mathcal{A}) \leq 2d\sqrt{\mathsf{Adv}^{\mathsf{GA\text{-}StCDH}}_{\mathsf{EGA}}(\mathcal{B})} + \frac{8(q+1)^2}{2^n} + \sqrt{\frac{32q_D(q_D+q)}{\sqrt{2^n}}} + \sqrt{\frac{4q_D}{2^n}} + \sqrt{\frac{40e^2(q+2q_D+2)^3}{2^n}},$$

and the running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the running time for using extractable random-oracle simulator for q_D extraction queries and q hash queries, which is about $\mathcal{O}(q \cdot q_D + q^2)$ and simulating H *for q queries, additionally $\mathcal{B}$ makes at most $q_D + p$ queries to its decision oracle.*

Note that n depends on the desired security level. Due to the fourth root term, n needs to be around four times the security parameter in bits. We discuss this in more detail in Section 6. We will now sketch the proof of Theorem 4. The full proof can be found in Appendix C.2.

Proof (Sketch). After some simple changes we first reprogram the random oracle H and G on the challenge inputs using O2H. Then the main idea of the proof is to simulate the random oracle G using the extractable random-oracle simulator from Definition 11. The reduction can then simulate decapsulation queries by extracting the inputs from the key-confirmation hash and verify the validity using the decision oracle $\mathsf{GA\text{-}DDH}(g \star \tilde{x}, \cdot, \cdot)$. Note that since the decapsulation oracle is classical, the extracted values are also classical and we only need classical access to $\mathsf{GA\text{-}DDH}(g \star \tilde{x}, \cdot, \cdot)$. Once we can simulate decapsulation without the secret key using the classical decision oracle, we can reduce the game to the GA-StCDH problem. ◻

4.3 Security of GA-HDH

The following theorem establishes security of GA-HDH based on the GA-DFQ-StCDH assumption. As opposed to the proof of GA-HEG, we have to use the semi-classical variant of the O2H lemma which yields a worse bound. We explain the reason in Appendix D.1.

Theorem 5. *For any quantum adversary $\mathcal{A}$ against the* CKS *security of* GA-HDH *that issues at most d parallel queries, each of size p, to the quantum-accessible random oracle* H*, there exists an adversary $\mathcal{B}$ against* GA-DFQ-StCDH *such that*

$$\mathsf{Adv}^{\mathsf{CKS}}_{\mathsf{GA\text{-}HDH}}(\mathcal{A}) \leq \sqrt{8(d+1)\mathsf{Adv}^{\mathsf{GA\text{-}DFQ\text{-}StCDH}}_{\mathsf{EGA}}(\mathcal{B})},$$

and the running time of $\mathcal{B}$ is about three times that of $\mathcal{A}$ plus $\mathcal{O}(q+p)$ queries to the decision oracle and the running time for simulating $\mathcal{O}(\max\{d \cdot p, q_R, q_T\})$ queries to the random oracle and $\mathcal{O}(q_O)$ rerandomizations on the set elements, where q_O, q_R and q_T are the number of register-honest, reveal and test queries.

We will only sketch the proof here. The full proof can be found in Appendix D.3.

Proof (Sketch). As in the proof of Theorem 3, our goal is to use a variant of the O2H lemma in order to randomize all challenge keys and bound the advantage of the O2H extractor using the GA-DFQ-StCDH assumption. However, instead of just a decapsulation oracle, we have to simulate the CorruptReveal oracle and the Test oracle. Although the adversary is allowed to choose identities for honest keys, we can compute all honest keys before the adversary can make any queries, so we can vary the behavior of the random oracle when it interacts with honest or corrupted keys. Note that this technique is not generally possible as the key generation could depend on the provided ID in other schemes. This allows to only hash $(\mathsf{ID}_1, \mathsf{ID}_1, \mathsf{pk}_1, \mathsf{pk}_2)$ without the shared DH value between pk_1 and pk_2, when at least one key is honest. Additionally, we can use a different internal random oracle, when both keys are honest. In the final reduction on GA-DFQ-StCDH, we embed the challenge set elements into the public keys using rerandomization. For each public key, we randomly choose which challenge element we use such that the adversary will issue a test query at least for one pair of identities containing both challenge elements. We can check whether quantum random oracle queries contain valid DH tuples using quantum access to the decision oracles. Then we can use the O2H lemma in its semi-classical variant and bound the success probability of its extractor with the GA-DFQ-StCDH assumption. ◻

5 Twinning for Group Actions

In this section, we adapt the twinning technique from [11] to the group actions setting. Due to the limited structure that group actions offer, we need a novel approach to develop and analyze the underlying trapdoor test. The trapdoor test will allow us to effectively simulate a decision oracle, apart from a small error probability. In contrast to the original twinning approach, the analysis of the error term is more involved and depends on an additional parameter m, which affects the "twinning factor". To illustrate this in an example: whereas in the traditional prime-order group setting, twinning doubles the size of public keys, the group action twinning technique will result in a public key of length m.

Using this technique we get two new schemes $\mathsf{GA\text{-}Twin\text{-}HEG}_m$ and $\mathsf{GA\text{-}Twin\text{-}HDH}_m$, the twinned versions of $\mathsf{GA\text{-}HEG}$ and $\mathsf{GA\text{-}HDH}$, which will be presented and analyzed in Sections 5.2 and 5.3. It allows us to remove the strong variants of $\mathsf{GA\text{-}CDH}$ including quantum access to decision oracles in the security proofs. Consequently we obtain a proof based on the standard $\mathsf{GA\text{-}CDH}$ assumption, albeit in exchange for larger keys and overall increased computation cost. Nevertheless, using our new twinning technique is thus far the only known method that allows for a security proof of a NIKE scheme from standard assumptions in the QROM. In Section 6 we discuss different parameter choices for m.

5.1 A Trapdoor Test

In order to replace the $\mathsf{GA\text{-}(FQ\text{-})StCDH}$ assumption, an algorithm must be able to simulate the decision oracle $\mathsf{GA\text{-}DDH}_g$ without knowing g explicitly. The following trapdoor test will be our basic tool to achieve this task.

Lemma 3 (Trapdoor Test). *Let $\mathsf{EGA} = (\mathcal{G}, \mathcal{X}, \star, \tilde{x})$, $\ell, m \in \mathbb{N}$ such that $1 < \ell < m/2$. Suppose $x_0, x_1, ..., x_{\ell-1}$, $s_\ell, ..., s_m$, $h_\ell, ..., h_m$ are mutually independent random variables, where $x_0, x_1, ..., x_{\ell-1}$ take values in $\mathcal{X}$, and for all $i \in [\ell, m]$ s_i are uniformly distributed over $[0, \ell-1]$ with the additional condition that each value in $[0, \ell-1]$ is taken at least once. Further, for all $i \in [\ell, m]$ h_i are uniformly distributed over $\mathcal{G}$. Define random variables $x_\ell, ..., x_m$, where $x_i = h_i \star x_{s_i}$ for $i \in [\ell, m]$. Further, let $g_i \in \mathcal{G}$ such that $x_i = g_i \star \tilde{x}$ for every $i \in [m]$. In addition, suppose that $\bar{z}_0, \bar{z}_1, ..., \bar{z}_m$ are random variables taking values in $\mathcal{X}$.*

We define

$$F_0(\bar{z}_0, \ldots, \bar{z}_m) \coloneqq \begin{cases} 1 & \text{if } \bar{z}_i = h_i \star \bar{z}_{s_i} \quad \forall i \in [\ell, m] \\ 0 & \text{else} \end{cases} \tag{4}$$

and

$$F_1(\bar{z}_0, \ldots, \bar{z}_m) \coloneqq \begin{cases} 1 & \text{if } \bar{z}_i = g_i \star \bar{z}_0 \quad \forall i \in [m] \\ 0 & \text{else} \end{cases} \tag{5}$$

and the advantage of an adversary $\mathcal{A}$ in distinguishing F_0 from F_1 with oracle access to one of the two functions and making at most q queries of depth d as

$$\mathsf{Adv}^{\mathsf{TDT}}_{\mathsf{EGA},q,d,\ell,m}(\mathcal{A}) \coloneqq \left|\Pr[\mathcal{A}^{F_0} \Rightarrow 1] - \Pr[\mathcal{A}^{F_1} \Rightarrow 1]\right|$$

We call eq. (4) the Trapdoor Test. *The following properties hold:*

1. *$x_\ell, ..., x_m$ are uniformly distributed over $\mathcal{X}$;*
2. *x_i and x_j are independent for all $i \in [0, \ell-1], j \in [\ell, m]$;*
3. *if $F_1(z) = 1$, then also $F_0(z) = 1$ for any input vector $\vec{z}$;*
4. *for any classical (quantum) adversary $\mathcal{A}$ with oracle access to F_b for $b \in \{0, 1\}$, the probability that $\mathcal{A}$ outputs 1 after at most q queries to F_b with query depth d is upper-bounded by the advantage of a classical (quantum) adversary $\mathcal{B}$ against the GDP problem for a function $T : \mathcal{Y} \to \{0, 1\}$ with $\Pr[T(x) = 1 \colon x \xleftarrow{\$} \mathcal{Y}] \le \frac{1}{|\mathcal{Y}|}$ and $|\mathcal{Y}| = \ell!\ell^{m-2\ell+1}$ (see Remark 4). Specifically,*

$$\mathsf{Adv}^{\mathsf{TDT}}_{\mathsf{EGA},q,d,\ell,m}(\mathcal{A}) \le \mathsf{Adv}^{\mathsf{GDP}}_{T,q,d}(\mathcal{B}) \le \begin{cases} \frac{2q}{|\mathcal{Y}|} & \text{(classical)} \\ 4\sqrt{\frac{(d+1)q}{|\mathcal{Y}|}} & \text{(quantum)} \end{cases}.$$

Adversary $\mathcal{B}^{T}$	**Oracle** $F(\bar{z}_0, \ldots, \bar{z}_m)$	**Function** $\mathrm{Convert}(z_0, \ldots, z_m)$
$x_0 := \tilde{x}$	**if** $\bar{z}_i = h_i \star \bar{z}_0$ **for** $i \in [m]$	**for** $i \in [\ell, m]$
for $i \in [m]$	**return** 1	**for** $j \in [0, \ell - 1]$
$h_i \xleftarrow{\$} \mathcal{G}$	$t := \mathrm{Convert}(\bar{z}_0, \ldots, \bar{z}_m)$	**if** $z_i = (h_i \cdot h_j) \star z_0$
$x_i := h_i \star \tilde{x}$	**return** $T(t)$	$s_i := j$
$b \leftarrow \mathcal{A}^{F}(x_0, \ldots, x_m)$		**return** $\mathrm{map}(s_\ell, \ldots, s_m)$
return b		

Fig. 11. Adversary $\mathcal{B}^{|T\rangle}$ against the GDP problem for the function T. The function "map" is the selected bijection from the set of possible s_i into $\mathcal{Y}$.

Proof. Properties 1. to 3. hold by inspection. For property 4., we build an adversary $\mathcal{B}$ on the GDP problem from a successful distinguisher $\mathcal{A}$ of the trapdoor test. The proofs are identical for the classical and quantum case as the oracles that $\mathcal{B}$ has to implement can all be defined as classical functions which make classical queries to other oracles, so by making all oracles quantum, the proof does not change.

First note that if $\mathcal{A}$ only queries tuples $z_0, \ldots, z_m$ to its function F_b for which x_i, z_0, z_i form a DH tuple, then both oracles always behave identically, so we assume that it will not make such queries. Since the s_i take all values in $[0, \ell - 1]$, for non-DH queries, the oracles differ only if $\mathcal{A}$ guesses *all* s_i used to generate the x_i correctly. In that case it could choose the first ℓ elements at random and set the last $m - \ell + 1$ elements to $g_i \star x_{s_i}$, where the g_i are the discrete logarithms of the i-th randomly chosen element. If the s_i do not cover all values in $[0, \ell - 1]$, this argument does not hold (see Remark 5).

We will construct an adversary $\mathcal{B}$ on the GDP problem for a function T, which will simulate the function F_1 if T is the all-zero function and F_0, i.e. the trapdoor test, if not. Specifically, let $T : \mathcal{Y} \to \{0, 1\}$ such that there is a bijective mapping from $\mathcal{Y}$ into the set of all possible combinations of s_i.

We describe $\mathcal{B}$ in Figure 11. First, $\mathcal{B}$ sets x_0 to the origin element $\tilde{x}$ and chooses m random elements $x_1, \ldots x_m$ and runs $\mathcal{A}$ on them as input. When $\mathcal{A}$ makes a query to F, $\mathcal{B}$ first checks if $\mathcal{A}$ provided a valid DH tuple and if so, returns 1. Otherwise, it computes which s_i were (implicitly) chosen to generate the query and maps them to the unique element they correspond to in $\mathcal{Y}$. Then it queries this element to its own function T and returns the result.

If T is the all-zero function, then F only returns 1 if the first check succeeds, i.e., F is equal to F_1 from eq. (5). Otherwise, there is exactly one entry in T for which it returns 1. Therefore, by returning the result of the query to T, $\mathcal{B}$ implicitly chooses its s_i as the ones corresponding to said entry in T and therefore simulates F_0 from eq. (4). So by outputting the same result as $\mathcal{A}$, $\mathcal{B}$ wins if and only if $\mathcal{A}$ wins and the claim follows. The quantum bound then follows directly from Lemma 2. □

Remark 4 (Sampling s_i). Let $\ell, m \in \mathbb{N}$ as in Lemma 3 and $k = m - \ell + 1$. Define

$$\mathcal{Y}^* = \{(s_\ell, \ldots, s_m) \in [0, \ell - 1]^k \mid \forall i \in [0, \ell - 1]\ \exists j : s_j = i\}.$$

In principal this is the set of possible values for the $(s_\ell, \ldots, s_m)$ from the lemma. The cardinality of $\mathcal{Y}^*$ may be described by the *Stirling partition number* multiplied by $\ell!$, more precisely

$$|\mathcal{Y}^*| = \ell! \cdot \left\{ \begin{matrix} k \\ \ell \end{matrix} \right\} = \sum_{i=0}^{d} (-1)^i \binom{\ell}{i} (\ell - i)^k.$$

One possibility to sample randomly from the entire set $\mathcal{Y}^*$ is rejection sampling from $[0, \ell - 1]^k$. Since this is not very practical, we suggest the following sampling method which samples from the strictly smaller subset $\mathcal{Y}$ of size $\ell! \ell^{k-\ell}$.

In order to ensure that the s_i take each value in $[0, \ell - 1]$, we first sample exactly these ℓ elements and then sample the remaining $k - \ell$ elements uniformly at random from $[0, \ell - 1]$.

Remark 5 (Necessity of the condition on s_i). The assumption that each value in $[0, \ell - 1]$ is taken at least once by the s_i is a necessary assumption. Otherwise, an adversary can simply guess a value $\alpha \in [0, \ell - 1]$ that is not taken by the s_i and subsequently choose $\bar{z}_\alpha$ randomly while computing all other $\bar{z}_i$ honestly. This would lead to

$$1 = F_0(\bar{z}_0, ..., \bar{z}_\alpha, ..., \bar{z}_m) \neq F_1(\bar{z}_0, ..., \bar{z}_\alpha, ..., \bar{z}_m) = 0$$

Gen	Encaps(pk)	Decaps(sk, ct)
$\mathsf{sk} := (h_1, ..., h_m) \xleftarrow{\$} \mathcal{G}^m$	$r \xleftarrow{\$} \mathcal{G}$	$K := \mathsf{H}(\mathsf{ct}, h_1 \star \mathsf{ct}, ..., h_m \star \mathsf{ct})$
$\mathsf{pk} := (y_1, ..., y_m) := (h_1 \star \tilde{x}, ...h_m \star \tilde{x})$	$\mathsf{ct} := r \star \tilde{x}$	**return** K
return $(\mathsf{pk}, \mathsf{sk})$	$K := \mathsf{H}(\mathsf{ct}, r \star y_1, ..., r \star y_m)$	
	return (ct, K)	

Fig. 12. Twin Hashed ElGamal KEM $\mathsf{GA\text{-}Twin\text{-}HEG}_m$ with twinning parameter m. $\mathsf{H} : \mathcal{X}^{m+1} \to \{0,1\}^\kappa$ is a hash function.

because $\bar{z}_\alpha$ is never used on the right side of $\bar{z}_i = h_i \star \bar{z}_{s_i}$ during the trapdoor test in (4). Therefore, the adversary is able to distinguish both functions without guessing all s_i which prevents the aforementioned reduction.

In order to use the trapdoor test in security proofs, we need to choose m and ℓ such that the advantage defined above becomes a small statistical factor. In Section 6, we compute these values for a security level of 128 bits.

5.2 Twin Hashed ElGamal

Applying the twinning technique to Hashed ElGamal yields the Twin Hashed ElGamal encryption scheme $\mathsf{GA\text{-}Twin\text{-}HEG}_m$ for an integer $m \in \mathbb{N}$, which is formally described in Figure 12. While twinning significantly increases the public key size and computation for both encapsulation and decapsulation, it allows us to prove its $\mathsf{IND\text{-}CCA}$ security without the use of strong variants of the $\mathsf{GA\text{-}CDH}$ problem. Furthermore, the ciphertext still consists of only one element.

Theorem 6. *Let $\ell, m \in \mathbb{N}$ such that $1 < \ell < m/2$. For any quantum adversary $\mathcal{A}$ against* $\mathsf{IND\text{-}CCA}$ *security of* $\mathsf{GA\text{-}Twin\text{-}HEG}_m$ *that issues at most q queries to the quantum-accessible random oracle* H *with query depth d, there exists a quantum adversary $\mathcal{B}$ against* $\mathsf{GA\text{-}CDH}$ *such that*

$$\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{GA\text{-}Twin\text{-}HEG}_m}(\mathcal{A}) \leq 4d\mathsf{Adv}^{\mathsf{GA\text{-}CDH}}_{\mathsf{EGA}}(\mathcal{B}) + 4\sqrt{\frac{(d+1)q}{\ell! \ell^{m-2\ell+1}}} \ ,$$

and the running time of $\mathcal{B}$ is about three times that of $\mathcal{A}$ plus the time to simulate $\mathcal{O}(\max\{q, q_D\})$ queries to H*, where q_D is the number of decapsulation queries.*

We will only sketch the proof here and refer to Appendix C.3 for the full proof. In fact, it is similar to the one of Theorem 3, only that we use the trapdoor test whenever the other proof uses the decision oracle.

Proof (Sketch). Let $\mathcal{A}$ be a quantum adversary in the $\mathsf{IND\text{-}CCA}$ game. Our goal is to construct an adversary $\mathcal{B}$ against $\mathsf{GA\text{-}CDH}$. The main question is how $\mathcal{B}$ simulates decapsulation queries. Therefore, let H_1 and H_2 be *internal* random oracles, the first is used for valid DH tuples and the second for invalid ones. Since for every ciphertext element x_1 there exists a unique vector of m set elements s.t. these form a DH tuple with the public key set elements, the output of H_1 only depends on x_1. We can check if a query consists of valid DH tuples using the trapdoor test. After this change, $\mathcal{B}$ can simulate decapsulation queries by just returning $\mathsf{H}_1(x_1)$. Next, we can apply the MRM-O2H lemma to reprogram H on the challenge ciphertext c^* and the corresponding DH tuples $(\mathsf{sk}[i] \star c^*)_{i \in [m]}$. For this the adversary $\mathcal{B}$ needs to be able to simulate H and H' (the reprogrammed H), which it can do using the trapdoor test. Note that since we applied the variant which considers parallel random oracle queries, the measured inputs are a set of size p. Due to the trapdoor test $\mathcal{B}$ can find the correct solution. In the final game, since the key K^* is now independent of the bit b, the adversary wins the game with probability $1/2$ and the claimed bound follows. □

5.3 Twin NIKE

We construct a NIKE scheme $\mathsf{GA\text{-}Twin\text{-}HDH}_m$ from an effective group action $\mathsf{EGA} = (\mathcal{G}, \mathcal{X}, \star, \tilde{x})$, which defines the public parameters pp together with an integer $m \in \mathbb{N}$ and a hash function $\mathsf{H} : \{0,1\}^* \to$

$\{0,1\}^\kappa$, thus defining $\mathcal{SHK} = \{0,1\}^\kappa$. As in Section 3, we assume that the identities can be represented by bitstrings of fixed length μ. On input an ID, the key generation algorithm chooses m group elements $(g_1, ..., g_m) \xleftarrow{\$} \mathcal{G}^m$ which form the secret key $\mathsf{sk}_{\mathsf{ID}}$. The public key is computed as $\mathsf{pk}_{\mathsf{ID}} = (g_1 \star \tilde{x}, ..., g_m \star \tilde{x}) \in \mathcal{X}^m$. The shared key of an identity ID_1 with public key $\mathsf{pk}_{\mathsf{ID}_1} = (x_1, ..., x_m)$ and an identity ID_2 with secret key $\mathsf{sk}_{\mathsf{ID}_2} = (g_1, ..., g_m)$ is defined as

$$K = \begin{cases} \mathsf{H}(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_{\mathsf{ID}_1}, \mathsf{pk}_{\mathsf{ID}_2}, g_1 \star x_1, ..., g_1 \star x_m, ..., g_m \star x_1, ..., g_m \star x_m) & \textbf{if } \mathsf{ID}_1 < \mathsf{ID}_2 \\ \mathsf{H}(\mathsf{ID}_2, \mathsf{ID}_1, \mathsf{pk}_{\mathsf{ID}_2}, \mathsf{pk}_{\mathsf{ID}_1}, g_1 \star x_1, ..., g_m \star x_1, ..., g_1 \star x_m, ..., g_m \star x_m) & \textbf{if } \mathsf{ID}_2 < \mathsf{ID}_1 \end{cases}$$

See Figure 13 for a schematic overview of our construction.

Again, twinning significantly increases the public key size and computation of $\mathsf{GA\text{-}Twin\text{-}HDH}_m$ compared to $\mathsf{GA\text{-}HDH}$, but allows us to use the same techniques as in Theorem 6 to prove security without relying on strong assumptions. This is formalized in Theorem 7.

Alice A	**Bob** B
$\mathsf{sk}_\mathsf{A} = (a_1, ..., a_m) \xleftarrow{\$} \mathcal{G}^m$	$\mathsf{sk}_\mathsf{B} = (b_1, ..., b_m) \xleftarrow{\$} \mathcal{G}^m$
$\mathsf{pk}_\mathsf{A} = (x_1^\mathsf{A}, ..., x_m^\mathsf{A}) = (a_1 \star \tilde{x}, ..., a_m \star \tilde{x})$	$\mathsf{pk}_\mathsf{B} = (x_1^\mathsf{B}, ..., x_m^\mathsf{B}) = (b_1 \star \tilde{x}, ..., b_m \star \tilde{x})$
for $i \in [m], j \in [m]$	**for** $i \in [m], j \in [m]$
$\quad z_{i,j} \coloneqq a_i \star x_j^\mathsf{B}$	$\quad z_{i,j} \coloneqq b_j \star x_i^\mathsf{A}$

$$K \coloneqq \mathsf{H}(\mathsf{A}, \mathsf{B}, \mathsf{pk}_\mathsf{A}, \mathsf{pk}_\mathsf{B}, z_{1,1}, ..., z_{1,m}, ..., z_{m,1}, ..., z_{m,m})$$

Fig. 13. Our NIKE Protocol $\mathsf{GA\text{-}Twin\text{-}HDH}_m$.

Theorem 7. *Let $\ell, m \in \mathbb{N}$ such that $1 < \ell < m/2$. For any quantum adversary $\mathcal{A}$ against the* CKS *security of* $\mathsf{GA\text{-}Twin\text{-}HDH}_m$ *that issues at most q queries to the quantum-accessible random oracle* H *of query depth d, there exists a quantum adversary $\mathcal{B}$ against* $\mathsf{GA\text{-}CDH}$ *such that*

$$\mathsf{Adv}^{\mathsf{CKS}}_{\mathsf{GA\text{-}Twin\text{-}HDH}_m}(\mathcal{A}) \leq \sqrt{8d\mathsf{Adv}^{\mathsf{GA\text{-}CDH}}_{\mathsf{EGA}}(\mathcal{B})} + 4\sqrt{\frac{(d+1)q}{\ell!\ell^{m-2\ell+1}}},$$

and the running time of $\mathcal{B}$ is about three times that of $\mathcal{A}$ plus the time needed to simulate $\mathcal{O}(\max\{q, q_R, q_T\})$ queries to the random oracle, to perform $\mathcal{O}(q_O)$ rerandomizations on set elements and to run the trapdoor test $\mathcal{O}(q)$ times, where q_O, q_R and q_T are the number of register-honest, reveal and test queries.

The proof is similar to the proof of Theorem 5 with the main difference that we use the trapdoor test whenever the other proof used the decision oracles. We defer the complete proof to Appendix D.4.

Proof (Sketch). As in the KEM proof, our goal is to use a variant of the O2H lemma in order to randomize all challenge keys and bound the advantage of the O2H extractor using the $\mathsf{GA\text{-}CDH}$ assumption. However, instead of just a decapsulation oracle, we have to simulate the CorruptReveal and Test oracles. Although the adversary is allowed to choose identities for honest keys, we can compute the public keys before it makes any queries, so we can vary the behavior of the random oracle when it interacts with honest or corrupted keys. Note that this technique is not generally possible as the key generation could depend on the provided ID in other schemes. This allows us to make similar conceptual changes as in the KEM proof, where we only hash $(\mathsf{ID}_1, \mathsf{ID}_1, \mathsf{pk}_1, \mathsf{pk}_2)$ without the $z_{i,j}$, when at least one key is honest. Additionally, we can use a different internal random oracle, when both keys are honest. By using the trapdoor test, we can remove the need for the secret keys completely. Finally, we can use the O2H lemma in its semi-classical variant and bound the success probability of its extractor with the $\mathsf{GA\text{-}CDH}$ assumption. For a discussion on why we cannot use the MRM variant, see Appendix D.1. □

Scheme	$\|\mathsf{pk}\|$	$\|\mathsf{ct}\|$	Gen	Encaps	Decaps	Assumption	Bound
GA-HEG (Fig. 3)	$\|\mathcal{X}\|$	$\|\mathcal{X}\|$	1	2	1	GA-FQ-StCDH	$d\,\mathsf{Adv}$
GA-HEG-KC (Fig. 10)	$\|\mathcal{X}\|$	$\|\mathcal{X}\| + 4\lambda$	1	2	1	GA-StCDH	$d\sqrt{\mathsf{Adv}}$
$\mathsf{GA\text{-}Twin\text{-}HEG}_m$ (Fig. 12)	$m \cdot \|\mathcal{X}\|$	$\|\mathcal{X}\|$	m	$m+1$	m	GA-CDH	$d\,\mathsf{Adv}$
GA-EG-FO [12,16]	$\|\mathcal{X}\|$	$\|\mathcal{X}\| + 2\lambda$	1	2	2	GA-CDH	$q\sqrt{\mathsf{Adv}}$
GA-EG-FO [12,31]	$\|\mathcal{X}\|$	$\|\mathcal{X}\| + 3\lambda$	1	2	2	GA-DDH	$d^2\mathsf{Adv}$
GA-HDH (Fig. 4)	$\|\mathcal{X}\|$	-	1	1 (SharedKey)		GA-DFQ-StCDH	$\sqrt{d\,\mathsf{Adv}}$
$\mathsf{GA\text{-}Twin\text{-}HDH}_m$ (Fig. 13)	m	-	m	m^2 (SharedKey)		GA-CDH	$\sqrt{d\,\mathsf{Adv}}$

Table 1. Overview of our different protocols and comparison to FO variants. By $|\mathcal{X}|$ we denote the length of a set element in bits. The columns "Gen", "Encaps" and "Decaps" state the number of group action evaluations that are needed in order to perform the corresponding algorithm. For NIKE schemes this refers to the SharedKey algorithm. Bounds are stated without statistical terms and q, d denote the number of random oracle queries and the query-depth. The security parameter is denoted by λ. For $\lambda = 128$ bit security, we need $m = 85$. For FO-EG we assume the implicit rejection variants.

6 Parameter Choices and Comparison

In order to compare the different schemes we need to elaborate on the parameter n, which is the bit length of the output of hash function G in the hashed ElGamal scheme with key confirmation, and the twinning parameter m. Both depend on the desired security level which is usually stated in bits. Taking the corresponding terms in the bounds of Theorems 4 and 6 into account, we determine the success ratio of an adversary $\mathcal{A}$. The success ratio of $\mathcal{A}$ is computed as its advantage $\epsilon_{\mathcal{A}}$ divided by its running time $t_{\mathcal{A}}$ [22]. For λ-bit security, we then require $\epsilon_{\mathcal{A}}/t_{\mathcal{A}} \leq 2^{-\lambda}$.

Key Confirmation. The output of the hash function G determines the length of the second ciphertext element. In order to determine the length, we analyze the statistical terms in Theorem 4. Note the one with the fourth root is the most dominating one. Thus, for λ-bit security, we need to set $n \approx 4\lambda$, where we assume $q_D \leq q \lessapprox t_{\mathcal{A}}$ and ignore additive constants.

Twinning. The efficiency of the Twin ElGamal encryption scheme $\mathsf{GA\text{-}Twin\text{-}HEG}_m$ and the Twin NIKE scheme $\mathsf{GA\text{-}Twin\text{-}HDH}_m$ depends on the twinning parameter m which directly translates to the length of the public key. The security level is determined by the value of $\ell!\ell^{m-2\ell+1}$, where $\ell \in [1, m/2]$ may be chosen arbitrarily. Note that ℓ only appears in the proofs of Theorem 6 and Theorem 7, hence it has no direct effect on the corresponding protocols.

Again, we only analyze the statistical term in the bound. For λ-bit security, we need

$$\frac{4}{t_{\mathcal{A}}} \cdot \sqrt{\frac{(d+1)q}{\ell!\ell^{m-2\ell+1}}} \leq 2^{-\lambda}.$$

Similar as before, we may assume that $d \leq q \lessapprox t_{\mathcal{A}}$, hence for an optimal success ratio an adversary would choose $d = q$. This means that we need to choose m large enough so that $\ell!\ell^{m-2\ell+1} \geq 2^{2\lambda+4}$ for some $\ell \in [1, m/2]$. As an example, for $\lambda = 128$, optimality is achieved by $m = 85$ (with $\ell = 17$).

Instantiation of the Group Action. Every set element $x \in \mathcal{X}$ is represented by a bitstring. In CSIDH the length of this bitstring is $\log(p)$, where the size of $\mathcal{X}$ is in $O(\sqrt{p})$. Choosing the correct parameter size for CSIDH is an actively discussed topic in the community. Castryck et al. [12] propose a 1792-bit prime p to achieve $\lambda = 128$ bit quantum security.

Comparison. Table 1 provides an overview of the schemes analyzed in this paper and a comparison to the ElGamal KEMs that can be obtained by the FO transform. The base scheme is the most efficient one, with one ciphertext element and two group action evaluations for Encaps. It also achieves the best QROM bound without any square root terms, but it relies on the strongest non-standard assumption. Hashed ElGamal with key confirmation has a slightly larger ciphertext and comes with a worse bound, however, it relies only on the GA-StCDH assumption. Since twinning cannot be done efficiently in the group action setting, the twinned version of hashed ElGamal is the least efficient in terms of public key size and group action computation. Nevertheless, the ciphertext still consists of only one set element and we get security based on the standard GA-CDH assumption. At this point we want to stress again that this seems the only way to construct an actively-secure NIKE based on a standard assumption. Otherwise, one has to rely on the assumption with a quantum-accessible decision oracle.

Acknowledgments

The work of Julien Duman was supported by the German Federal Ministry of Education and Research (BMBF) in the course of the 6GEM Research Hub under Grant 16KISK037. Dominik Hartmann was supported by the BMBF iBlockchain project. Eike Kiltz was supported by the BMBF iBlockchain project, the Deutsche Forschungsgemeinschaft (DFG, German research Foundation) as part of the Excellence Strategy of the German Federal and State Governments – EXC 2092 CASA - 390781972, and by the European Union (ERC AdG REWORC - 101054911). Sabrina Kunzweiler, Jonas Lehmann and Doreen Riepel were funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (Apr 2001). https://doi.org/10.1007/3-540-45353-9_12
2. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 411–439. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_14
3. Alwen, J., Blanchet, B., Hauck, E., Kiltz, E., Lipp, B., Riepel, D.: Analysing the HPKE standard. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 87–116. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_4
4. Alwen, J., Hartmann, D., Kiltz, E., Mularczyk, M.: Server-aided continuous group key agreement. Cryptology ePrint Archive, Report 2021/1456 (2021), https://eprint.iacr.org/2021/1456
5. Ambainis, A., Hamburg, M., Unruh, D.: Quantum security proofs using semi-classical oracles. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 269–295. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_10
6. Barnes, R., Millican, J., Omara, E., Cohn-Gordon, K., Robert, R.: Message layer security (mls) wg. https://datatracker.ietf.org/wg/mls/about/
7. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 62–73. ACM Press (Nov 1993). https://doi.org/10.1145/168588.168596
8. Bindel, N., Hamburg, M., Hövelmanns, K., Hülsing, A., Persichetti, E.: Tighter proofs of CCA security in the quantum random oracle model. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 61–90. Springer, Heidelberg (Dec 2019). https://doi.org/10.1007/978-3-030-36033-7_3
9. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0_3
10. Boneh, D., Zhandry, M.: Secure signatures and chosen ciphertext security in a quantum computing world. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 361–379. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_21
11. Cash, D., Kiltz, E., Shoup, V.: The twin Diffie-Hellman problem and applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_8
12. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03332-3_15
13. Cohn-Gordon, K., Cremers, C., Gjøsteen, K., Jacobsen, H., Jager, T.: Highly efficient key exchange protocols with optimal tightness. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 767–797. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26954-8_25

14. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
15. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976)
16. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Online-extractability in the quantum random-oracle model. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022. pp. 677–706. Springer International Publishing, Cham (2022)
17. Duman, J., Hövelmanns, K., Kiltz, E., Lyubashevsky, V., Seiler, G., Unruh, D.: A thorough treatment of highly-efficient NTRU instantiations. Cryptology ePrint Archive, Report 2021/1352 (2021), https://eprint.iacr.org/2021/1352
18. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**, 469–472 (1985)
19. Fouotsa, T.B., Petit, C.: Sims: a simplification of sigamal. In: International Conference on Post-Quantum Cryptography. pp. 277–295. Springer (2021)
20. Freire, E.S.V., Hofheinz, D., Kiltz, E., Paterson, K.G.: Non-interactive key exchange. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 254–271. Springer, Heidelberg (Feb / Mar 2013). https://doi.org/10.1007/978-3-642-36362-7_17
21. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_34
22. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing **28**(4), 1364–1396 (1999)
23. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_12
24. Hövelmanns, K., Kiltz, E., Schäge, S., Unruh, D.: Generic authenticated key exchange in the quantum random oracle model. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 389–422. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45388-6_14
25. Hülsing, A., Ning, K.C., Schwabe, P., Weber, F., Zimmermann, P.R.: Post-quantum WireGuard. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 304–321. IEEE Computer Society (2021), https://cryptojedi.org/papers/#pqwireguard
26. Hülsing, A., Rijneveld, J., Song, F.: Mitigating multi-target attacks in hash-based signatures. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) PKC 2016, Part I. LNCS, vol. 9614, pp. 387–416. Springer, Heidelberg (Mar 2016). https://doi.org/10.1007/978-3-662-49384-7_15
27. Jiang, H., Zhang, Z., Ma, Z.: On the non-tightness of measurement-based reductions for key encapsulation mechanism in the quantum random oracle model. Cryptology ePrint Archive, Report 2019/494 (2019), https://eprint.iacr.org/2019/494
28. Kawashima, T., Takashima, K., Aikawa, Y., Takagi, T.: An efficient authenticated key exchange from random self-reducibility on CSIDH. In: Hong, D. (ed.) ICISC 20. LNCS, vol. 12593, pp. 58–84. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-68890-5_4
29. de Kock, B., Gjøsteen, K., Veroni, M.: Practical isogeny-based key-exchange with optimal tightness. In: Dunkelman, O., Jacobson, Jr., M.J., O'Flynn, C. (eds.) Selected Areas in Cryptography. pp. 451–479. Springer International Publishing, Cham (2021)
30. Krawczyk, H., Wee, H.: The optls protocol and tls 1.3. In: 2016 IEEE European Symposium on Security and Privacy (EuroS P). pp. 81–96 (2016). https://doi.org/10.1109/EuroSP.2016.18
31. Kuchta, V., Sakzad, A., Stehlé, D., Steinfeld, R., Sun, S.: Measure-rewind-measure: Tighter quantum random oracle model proofs for one-way to hiding and CCA security. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 703–728. Springer, Heidelberg (May 2020). https://doi.org/10.1007/978-3-030-45727-3_24
32. Moriya, T., Onuki, H., Takagi, T.: SiGamal: A supersingular isogeny-based PKE and its application to a PRF. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 551–580. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_19
33. Rescorla, E., Oku, K., Sullivan, N., Wood, C.A.: TLS Encrypted Client Hello. Internet-Draft draft-ietf-tls-esni-13, Internet Engineering Task Force (Aug 2021), https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-13
34. Rescorla, E., Sullivan, N., Wood, C.A.: Semi-Static Diffie-Hellman Key Establishment for TLS 1.3. Internet-Draft draft-rescorla-tls-semistatic-dh-02, Internet Engineering Task Force (Nov 2019), https://datatracker.ietf.org/doc/html/draft-rescorla-tls-semistatic-dh-02, work in Progress
35. Rostovtsev, A., Stolbunov, A.: Public-Key Cryptosystem Based On Isogenies. Cryptology ePrint Archive, Report 2006/145 (2006), https://eprint.iacr.org/2006/145

36. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1461–1480. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3423350
37. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th FOCS. pp. 124–134. IEEE Computer Society Press (Nov 1994). https://doi.org/10.1109/SFCS.1994.365700
38. Unruh, D.: Revocable quantum timed-release encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 129–146. Springer, Heidelberg (May 2014). https://doi.org/10.1007/978-3-642-55220-5_8
39. Yoneyama, K.: Post-quantum variants of ISO/IEC standards: Compact chosen ciphertext secure key encapsulation mechanism from isogeny. In: Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop. p. 13–21. SSR'19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3338500.3360336
40. Zhandry, M.: How to construct quantum random functions. In: 53rd FOCS. pp. 679–687. IEEE Computer Society Press (Oct 2012). https://doi.org/10.1109/FOCS.2012.37
41. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 758–775. Springer, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_44
42. Zhandry, M.: How to record quantum queries, and applications to quantum indifferentiability. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 239–268. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_9

A CSIDH: An Isogeny-based REGA

An important instantiation of REGAs is provided by isogeny-based group actions, in particular by CSIDH.

Let p be a large prime of the form $p = 4 \cdot \ell_1 \cdots \ell_n - 1$, where the ℓ_i are small distinct odd primes. Fix the elliptic curve $E_0 : y^2 = x^3 + x$ over $\mathbb{F}_p$. The curve E_0 is supersingular and its $\mathbb{F}_p$-rational endomorphism ring is $\mathcal{O} = \mathbb{Z}[\pi]$, where π is the Frobenius endomorphism. Let $\mathcal{E}\ell\ell_p(\mathcal{O})$ be the set of elliptic curves defined over $\mathbb{F}_p$, with endomorphism ring $\mathcal{O}$. The ideal class group $cl(\mathcal{O})$ acts on the set $\mathcal{E}\ell\ell_p(\mathcal{O})$, i.e. there is a map

$$\begin{aligned} \star : cl(\mathcal{O}) \times \mathcal{E}\ell\ell_p(\mathcal{O}) &\to \mathcal{E}\ell\ell_p(\mathcal{O}) \\ ([\mathfrak{a}], E) &\mapsto [\mathfrak{a}] \star E, \end{aligned}$$

satisfying the properties from Definition 3 [12, Theorem 7]. Moreover the analysis in [12] readily shows that $(cl(\mathcal{O}), \mathcal{E}\ell\ell_p(\mathcal{O}), \star, E_0)$ is indeed a REGA.

Remark 6. As pointed in the original paper [12], an inherent property of the CSIDH group is that given $E = [\mathfrak{a}] \star E_0 \in \mathcal{E}\ell\ell_p(\mathcal{O})$, one can efficiently compute the quadratic twist $E^t = [\mathfrak{a}]^{-1} \star E_0$. This property is not described in the cryptographic group action framework from [2]. However this should not affect the security of our protocols, since we only rely on variants of the standard group action computational Diffie-Hellman problem and (so far) no attacks using twists are known on this assumption.

B Quantum Preliminaries

We recall some quantum computation preliminaries as stated in [17].

Qubit. A qubit $|x\rangle = \alpha|0\rangle + \beta|1\rangle$ is a 2-dimensional unit vector with coefficients in $\mathbb{C}$, i.e. $x = (\alpha, \beta) \in \mathbb{C}^2$ fulfilling the normalization constraint $|\alpha|^2 + |\beta|^2 = 1$. When neither $\alpha = 1$ nor $\beta = 1$, we say that $|x\rangle$ is in *superposition.*

n-qubit state. An n-bit quantum register $|x\rangle = \sum_{i=1}^{2^n-1} \alpha_i |i\rangle$ is a unit vector of $\mathbb{C}^{2^n} = (\mathbb{C}^2)^{\otimes n}$, that is $\alpha_i \in \mathbb{C}$ and $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$. We call the set $\{|0\rangle, |1\rangle, \ldots, |2^n - 1\rangle\}$ the *computational basis.* When $|x\rangle$ can not be written as the tensor product of single qubits, we say that $|x\rangle$ is *entangled.*

Measurement. Unless otherwise stated, measurements are done in the computational basis. After measuring a quantum register $|x\rangle = \sum_{i=0}^{2^n-1} a_i |i\rangle$ in the computational basis, the state *collapses* and $|x\rangle = \pm |i\rangle$ with probability $|\alpha_i|^2$.

QUANTUM ALGORITHMS. A quantum algorithm $\mathcal{A}$ is a series of unitary operations U_i, where unitary operations are defined as to map unit vectors to unit vectors, preserving the normalization constraint of quantum registers. A quantum oracle algorithm $\mathcal{A}^{\mathrm{O}}$ is defined similarly, except it can query the oracle O after (or before) executing a unitary U_i. Since quantum computation needs to be reversible, we model an oracle $\mathrm{O} : X \to Y$ by a unitary U_{O} that maps $|x\rangle\,|y\rangle \mapsto |x\rangle\,|y \oplus \mathrm{O}(x)\rangle$.

QUANTUM RANDOM ORACLE MODEL. Following [9], we model quantum adversaries to have quantum access to random oracles since quantum adversaries can evaluate hash functions in superposition.

SIMULATING QUANTUM RANDOM ORACLES. We can simulate quantum random oracles either by 2q-wise independent functions [41] or using Zhandry's Compressed Oracle technique [42]. The former is only perfectly indistinguishable for up to q RO queries, while being conceptually simpler to understand. The later has the advantage of not requiring an upper bound on q, with the disadvantage of being more inaccessible for readers unfamiliar with the technique. For simplicity, the reader can imagine to instantiate the reductions using the former, while for the theorems we use the later technique.

QUANTUM-ACCESS OF ORACLES For an oracle O, we are going to write $|\mathrm{O}\rangle$ to denote that an algorithm has quantum-access on all inputs and O if it has not (which means that its inputs are implicitly measured). For an oracle which allows partial quantum-access, we write $|\cdot\rangle$ to denote the inputs which are quantum (i.e., not measured), for example $\mathrm{O}(\cdot, |\cdot\rangle)$ means that the first input is classical (i.e., implicitly measured on query) and the second is quantum. Alternatively to $|\mathrm{O}\rangle$ we might also write $\mathrm{O}(|\cdot\rangle, |\cdot\rangle)$, if O takes two inputs. Since all our proofs are in the QROM, it is clear that the adversary has quantum-access to its random oracles. Thus, we just write H instead of $|\mathsf{H}\rangle$ for a random oracle H.

QUERY DEPTH AND QUERY PARALLELISM. Following [5] we are going to consider the query depth d of an adversary making in total q random oracle queries. This is important in practice since for highly-parallel adversaries we have $d \ll q$. We obtain the bounds for sequential adversaries by setting $d := q$.

B.1 Oneway-to-Hiding Lemmas

Below, we recall the oneway-to-hiding lemma, which is used to reprogram random oracle values. Informally, Theorem 8 states that if a random oracle is reprogrammed on a set $\mathcal{S} \subset \mathcal{R}$ of inputs, the probability of an adversary $\mathcal{A}$ behaving differently can be related to the success probability of an extractor algorithm $\mathcal{B}$ which extracts at least one element of $\mathcal{S}$ by measuring the query register of one of $\mathcal{A}$'s randomly chosen oracle queries.

Theorem 8 (Original O2H, Theorem 3 from the eprint version of [5]). *Let $\mathcal{S} \subset \mathcal{R}$ be random. Let G, H be random functions satisfying $\forall r \notin \mathcal{S} : \mathsf{G}(r) = \mathsf{H}(r)$. Let z be a random classical value. ($\mathcal{S}, \mathsf{G}, \mathsf{H}, z$ may have arbitrary joint distribution.) Let $\mathcal{A}$ be a quantum oracle algorithm with query depth d, expecting input z. Let Ext be the algorithm which on input z samples a uniform i from $\{1, \ldots, d\}$, runs $\mathcal{A}$ right before its ith query to G, measures all query input registers and outputs the set $\mathcal{T}$ of measurement outcomes. Then*

$$\left|\Pr[\mathcal{A}^{\mathsf{G}}(z) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{H}}(z) \Rightarrow 1]\right| \le 2d\sqrt{\Pr[\mathcal{S} \cap \mathcal{T} \neq \varnothing : \mathcal{T} \leftarrow \mathsf{Ext}^{\mathsf{H}}(z)]}.$$

Remark 7. As explained in [5] the O2H theorems also hold if the adversary $\mathcal{A}$ has access to additional oracles, since those can be encoded using z. For improved readability we are still going to write down the additional oracles as a superscript, that is $\mathcal{A}^{\mathsf{H},\mathrm{O}} = \mathcal{A}^{\mathsf{H}}(\mathrm{O})$ and $\mathsf{Ext}^{\mathsf{H},\mathrm{O}} = \mathsf{Ext}^{\mathsf{H}}(\mathrm{O})$ and reserve z for the non-oracle inputs.

Definition 8 (Unitary Quantum Oracle Algorithm). *Let $q \in \mathbb{N}$. An algorithm $\mathcal{A}^{\mathrm{O}}$ is called a* unitary quantum oracle algorithm *with query depth d and query parallelism p, if there are unitaries $U_1, \ldots, U_{q+1}$ such that $\mathcal{A}$'s execution can be described as*

$$\mathcal{A} = U_{d+1} \circ \mathrm{O}^{\otimes p} \circ U_d \circ \ldots \circ \mathrm{O}^{\otimes p} \circ U_1,$$

where $\mathcal{A}$'s output is defined as the measurement of its quantum state in the standard basis after applying U_{d+1}. For multiple oracles it is defined analogously.

Remark 8. At this point we want to highlight that we state the MRM O2H lemma (Lemma 1) slightly different than in [31], Lemma 3.3, by limiting it to unitary/reversible algorithms. The reason is that the property is used in the proof in order to rewind the adversary. Note that non-reversible quantum algorithms can be efficiently turned into reversible quantum algorithms, using the deferred measurement principle. One subtlety though is that if the quantum algorithm has access to *classical* oracles, which implicitly measure their input, then the new reversible adversary has to get quantum-access to the oracle, since the measurement is deferred to the end of the algorithm. Hence, the reduction needs to be able to simulate this, once classical oracle, on quantum superpositions. This was observed by [27]. In the proofs where we apply the MRM O2H lemma, we explain how to simulate quantum decapsulation queries, which is why we can apply the MRM O2H lemma. In the CKS security model there are more oracles available and it hence appears more difficult to show how to simulate them all quantumly, we dicuss this difficulty in Appendix D.1. For a more thorough discussion of this issue, see [27].

Definition 9 (Semi-Classical and Punctured Oracles). *Let $f : \mathcal{X} \to \{0,1\}$ be a function and $\mathcal{S} \subset \mathcal{X}$ a subset s.t. $f(x) = 1$ if and only if $x \in S$. The semi-classical oracle O_f^{SC} (or equivalently O_S^{SC}) is defined as the composition of the unitary U_f and a measurement of the output register in the standard basis.*

Let $\mathcal{A}^{\mathrm{O}_f^{SC}}(z)$ be a quantum algorithm which gets arbitrary input z and access to O_f^{SC}. We call the event that $\mathcal{A}$ queries O_f^{SC} on an input that yields 1 FIND.

Let H be another quantum oracle with domain $\mathcal{X}$ and some codomain Y. We define the punctured *oracle $H \setminus \mathcal{S}$ as a quantum oracle that first runs O_S^{SC} and then H on the result.*

Note that as long as FIND does not occur, H and $H \setminus \mathcal{S}$ behave identically.

We now recall the semi-classical variant of the oneway-to-hiding lemma from [5].

Theorem 9 (Semi-Classical O2H ([5], Theorem 1)). *Let $\mathcal{S} \subset \mathcal{R}$ be random. Let G, H be random functions satisfying $\forall r \notin \mathcal{S} : \mathsf{G}(r) = \mathsf{H}(r)$. Let z be a random classical value. ($\mathcal{S}, \mathsf{G}, \mathsf{H}, z$ may have arbitrary joint distribution.) Let $\mathcal{A}$ be a quantum oracle algorithm with query depth d, expecting input z and*

$$P_{left} := \Pr\left[b = 1 \mid b \leftarrow \mathcal{A}^{\mathsf{G}}(z)\right] \; P_{right} := \Pr\left[b = 1 \mid b \leftarrow \mathcal{A}^{\mathsf{H}}(z)\right] \; P_{find} := \Pr\left[\textit{FIND} \;\middle|\; \mathcal{A}^{\mathsf{G}\setminus\mathcal{S}}\right]$$

Then

$$|P_{left} - P_{right}| \leq 2\sqrt{(d+1)P_{Find}} \tag{6}$$

and

$$|\sqrt{P_{left}} - \sqrt{P_{right}}| \leq 2\sqrt{(d+1)P_{Find}}\ . \tag{7}$$

We also recall a bound on the probability of FIND occurring in this setting.

Theorem 10 (Search in Semi-Classical Oracles ([5], Theorem 2 and Corollary 1)). *Let $\mathcal{S} \subset \mathcal{R}$ be random. Let G be a semi-classical oracle with domain R. Let z be a random classical value. ($\mathcal{S}, \mathsf{G}, \mathsf{H}, z$ may have arbitrary joint distribution.) Let $\mathcal{A}$ be a quantum oracle algorithm with query depth d, expecting input z. Let Ext be the algorithm which on input z samples a uniform i from $\{1, \ldots, d\}$, runs $\mathcal{A}^{\mathrm{O}_\varnothing^{SC}}$ right before its ith query to G, measures all query input registers and outputs the set $\mathcal{T}$ of measurement outcomes. Then*

$$\Pr[\textit{FIND} \mid \mathcal{A}^{\mathrm{O}_\mathcal{S}^{SC}}(z)] \leq 4d \Pr[\mathcal{S} \cap \mathcal{T} \neq \varnothing \mid \mathcal{T} \leftarrow \mathsf{Ext}^{\mathsf{H}}(z)]\,.$$

Moreover, if $\mathcal{S}$ and z are independent and $\mathcal{A}$ is a q-query algorithm we have

$$\Pr[\textit{FIND} \mid \mathcal{A}^{\mathrm{O}_\mathcal{S}^{SC}}(z)] \leq 4q \cdot P_{max}\,, \tag{8}$$

where $P_{max} := \max_{x\in\mathcal{X}} \Pr[x \in \mathcal{S}]$.

B.2 Proof of Lemma 2

Proof (Lemma 2). The proof is a simple consequence of the semi-classical O2H lemma, by just reprogramming the elements where F and the zero function K differ. In the search variant of the proof, the authors of [5] could apply eq. (7), since the right-hand side of it vanishes, which yields a quadratically better bound. In the decision variant the right-hand side does not vanish, so we use eq. (6) instead. Let $\mathcal{A}$ be a q-query algorithm of depth d.

Game G_1. We define $\mathrm{G}_1^{\mathcal{A}}$ to be $\mathsf{GDP}_{\mathsf{F},1}^{\mathcal{A}}$. By definition,

$$\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathsf{GDP}_{\mathsf{F},1}^{\mathcal{A}} \Rightarrow 1]\,.$$

Game G_2. We reprogramm F on all elements which map to 1 to now map to 0, that is we define $\mathrm{G}_2^{\mathcal{A}}$ to be $\mathsf{GDP}_{\mathsf{F},0}^{\mathcal{A}}$ and let $\mathcal{S} := \{x \in \mathcal{X} \mid \mathsf{F}(x) = 1\}$. By eq. (6) of Theorem 9, we have

$$\left|\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1]\right| \leq 2\sqrt{(d+1)\Pr\left[\textsc{Find} \mid \mathcal{A}^{\mathsf{F}\setminus\mathcal{S}}\right]}\,. \tag{9}$$

We bound the right-hand probability using eq. (8), we have

$$\Pr\left[\textsc{Find} \,\middle|\, \mathcal{A}^{\mathsf{F}\setminus\mathcal{S}}\right] \leq 4q \cdot P_{\max}\,, \tag{10}$$

where

$$P_{\max} := \max_{x \in \mathcal{X}} \Pr[x \in \mathcal{S}] \leq \lambda\,, \tag{11}$$

by definition of F. Bounding eq. (10) by eq. (11) and eq. (9) by eq. (10) and moving 4 outside of the square-root yields

$$\left|\Pr[\mathsf{GDP}_{\mathsf{F},0}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{GDP}_{\mathsf{F},1}^{\mathcal{A}} \Rightarrow 1]\right| \leq 4\sqrt{(d+1)q\lambda}\,,$$

which concludes our proof. □

C Omitted Proofs for Hashed ElGamal Variants

In this section we provide the proofs for the hashed ElGamal variants $\mathsf{GA\text{-}HEG\text{-}KC}$ and $\mathsf{GA\text{-}Twin\text{-}HEG}_m$. Therefore, we first recall the definition of an extractable quantum random oracle simulator which we will need for the proof of $\mathsf{GA\text{-}HEG\text{-}KC}$.

C.1 Extractable Quantum Random Oracle Simulation

We recall a technical tool for the proof of Theorem 4 which was presented in [16]. Informally, the extractable random oracle simulator from [16], which uses Zhandrys Compressed Oracle technique [42], allows to extract the preimages x of a random oracle output y if one has access to a "commitment" t on y[4] similarly to the classical random oracle. Examples for such "commitments" are the hash image y itself or an encryption t of same value using y as its randomness (as long as the encryption scheme is sufficiently spread). Note that this technique is weaker than the full *preimage awareness* of classical random oracles, which can always output the preimage of any hash query on the fly, because a commitment is needed to extract the preimage. For a more in depth explanation we refer to [16,42].

Definition 10. *Let $n \in \mathbb{N}$, $\mathcal{X}, \mathcal{T}$ two sets, $f : \mathcal{X} \times \{0,1\}^n \to \mathcal{T}$ a function and $R \subset \mathcal{X} \times \{0,1\}^n$ a relation. We define*

$$\Gamma(f) := \max_{\substack{x \in \mathcal{X}\\ t \in \mathcal{T}}} |\{y \mid f(x,y) = t\}|,$$

$$\Gamma'(f) := \max_{\substack{x,x' \in \mathcal{X}\\ y' \in \{0,1\}^n}} |\{y \mid f(x,y) = f(x',y')\}|$$

[4] Strictly speaking, t does not need to be a commitment, only sufficiently determine y, but we follow the intuitive terminology of [16].

and

$$\Gamma_R := \max_{x \in \mathcal{X}} |\{y \mid (x,y) \in R\}| \ .$$

Definition 11 (Extractable Quantum Random Oracle Simulator). *Let $n \in \mathbb{N}$, $\mathcal{X}, \mathcal{T}$ two sets, $f : \mathcal{X} \times \{0,1\}^n \to \mathcal{T}$ a function and $R' \subset \mathcal{X} \times \mathcal{T}$ and $R \subset \mathcal{X} \times \{0,1\}^n$ relations with $(x,y) \in R \Leftrightarrow (x, f(x,y)) \in R'$.*

We define the stateful quantum simulator $\mathcal{S}$ *that has the (quantum-accessible) interfaces $\mathcal{S}.RO : \mathcal{X} \to \{0,1\}^n$ and $\mathcal{S}.E : \mathcal{T} \to \mathcal{X} \cup \perp$ and the following properties:*

1. *If no query to $\mathcal{S}.E$ is made, $\mathcal{S}.RO$ is indistinguishable from a (quantum) random oracle.*
2. *Any two independent queries to $\mathcal{S}.RO$ (resp. $\mathcal{S}.E$) commute.*
3. *Any two subsequent queries to $\mathcal{S}.E$ and $\mathcal{S}.RO$ $8\sqrt{\frac{\Gamma(f)}{2^{n-1}}}$-almost-commute.*
4. *Any query to $\mathcal{S}.RO$ (resp. $\mathcal{S}.E$) is idempotent, i.e. returns the same result if no other query was made in between.*
5. *If $\hat{x} = \mathcal{S}.E(t)$ and $\hat{h} = \mathcal{S}.RO(\hat{x})$ are two subsequent classical queries, then*
$$\Pr\left[\hat{x} \neq \perp \wedge f(\hat{x}, \hat{h}) \neq t\right] \leq \frac{2\Gamma(f)}{2^n} \ .$$
6. *If $h = \mathcal{S}.RO(x)$ and $\hat{x} = \mathcal{S}.E(f(x,h))$ are two subsequent classical queries, then*
$$\Pr[\hat{x} = \perp] \leq \frac{1}{2^{n-1}} \ .$$
7. *Let $\mathcal{A}$ be an adversary making at most q queries to the $\mathcal{S}.RO$ oracle and no queries to the $\mathcal{S}.E$ oracle, which outputs $t \in \mathcal{T}$. Then*
$$\Pr_{\substack{t \leftarrow \mathcal{A}^{\mathcal{S}.RO} \\ \hat{x} \leftarrow \mathcal{S}.E(t)}}[(\hat{x}, t) \in R'] \leq 128 \cdot q^2 \frac{\Gamma_R}{2^n} \ .$$
8. *Let $\mathcal{A}$ be an adversary making at most q queries to $\mathcal{S}.RO$ and no queries to $\mathcal{S}.E$, that outputs ℓ-tuples $(x_1, t_1), \ldots, (x_\ell, t_\ell) \in (\mathcal{X} \times \mathcal{T})$. Then*
$$\Pr_{\substack{(t_1,x_1),\ldots,(t_\ell,x_\ell) \leftarrow \mathcal{A}^{\mathcal{S}.RO} \\ h_1 \leftarrow \mathcal{S}.RO(x_1),\ldots,h_\ell \leftarrow \mathcal{S}.RO(x_\ell) \\ \hat{x}_1 \leftarrow \mathcal{S}.E(t_1),\ldots,\hat{x}_\ell \leftarrow \mathcal{S}.E(t_\ell)}} [\exists i \colon \hat{x}_i \neq x_i \wedge f(x_i, h_i) = t_i] \leq 40e^2 \cdot (q + \ell + 2)^3 \frac{\Gamma'(f)}{2^n} \ .$$
9. *The running time for $\mathcal{S}$ is bounded as $\mathcal{O}(q_{\mathsf{RO}} \cdot q_{\mathsf{E}} \cdot \mathsf{Time}(f) + q_{\mathsf{RO}}^2)$, where q_{E} is the numbers of queries to $\mathcal{S}.E$ and q_{RO} the number of queries to $\mathcal{S}.RO$.*

The existence of a simulator $\mathcal{S}$ as defined in Definition 11 was shown in [16].

Let us give some intuition on the properties of $\mathcal{S}$. Properties 1 and 2 ensure, that $\mathcal{S}$ behaves like a regular quantum random oracle, unless $\mathcal{S}.E$ is called *and* queries are dependent on one another. Property 3 tells us that extraction is only causes detectable change in the state of $\mathcal{S}$ with low probability (as long as f is sparse). Properties 5 and 6 state that extraction always works, if the preimage was queried before and that it may fail otherwise. Property 7 provides a bound on this failure probability, i.e. if one tries to extract a preimage to a value not queried yet, $\mathcal{S}.E$ will output $\perp$ with high probability. Lastly, Property 8 tells us that $\mathcal{S}.E$ cannot be used to find collisions, i.e. with high probability, it will give the same preimage that was used before.

C.2 Proof of Theorem 4

Proof. Let $\mathcal{A}$ be a quantum adversary as described in Theorem 4. Consider the games given in Figure 14. We proceed by analyzing the sequence of games.

GAME G_1. This is the $\mathsf{IND\text{-}CCA}$ game where we unfolded the definition of $\mathsf{GA\text{-}HEG\text{-}KC}$. By definition,

$$\left|\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] - \tfrac{1}{2}\right| = \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{GA\text{-}HEG\text{-}KC}}(\mathcal{A}) \,.$$

GAME G_2. In this game we return $\perp$ when the decapsulation oracle is queried on (c, d^*) for arbitrary c. This change is only noticeable to an adversary who finds a collision of the form $\mathsf{G}(c, g \star c) = d^*$ in G (with

Games G_1-G_{14}

$\mathsf{sk} := g \xleftarrow{\$} \mathcal{G}$
$\mathsf{pk} := x := g \star \tilde{x}$
$b \xleftarrow{\$} \{0,1\}$
$r \xleftarrow{\$} \mathcal{G}$
$c^* := r \star \tilde{x}$
$d^* := \mathsf{G}(c^*, r \star \mathsf{pk})$
$d^* \xleftarrow{\$} \{0,1\}^n$ $\quad\backslash\backslash\ \mathrm{G}_4$-$\mathrm{G}_{14}$
$K_0 := \mathsf{H}(c^*, r \star \mathsf{pk})$
$K_0 \xleftarrow{\$} \{0,1\}^\kappa$ $\quad\backslash\backslash\ \mathrm{G}_4$-$\mathrm{G}_{14}$
$K_1 \xleftarrow{\$} \{0,1\}^\kappa$
$b' \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{G},\textsc{Decaps}}(\mathsf{pk}, (c^*, d^*), K_b)$ $\quad\backslash\backslash\ \mathrm{G}_1$-$\mathrm{G}_4$
return $[\![b = b']\!]$ $\quad\backslash\backslash\ \mathrm{G}_1$-$\mathrm{G}_4$
$\mathcal{T} \leftarrow \mathsf{Ext}\mathcal{A}^{\mathsf{H},\mathsf{G},\textsc{Decaps}}(\mathsf{pk}, (c^*, d^*), K_b)$ $\quad\backslash\backslash\ \mathrm{G}_5$
$\mathcal{T} \leftarrow \mathsf{Ext}^{\mathsf{H},\mathcal{S}.RO,\textsc{Decaps}}(\mathsf{pk}, (c^*, d^*), K_b)$ $\quad\backslash\backslash\ \mathrm{G}_6$-$\mathrm{G}_{14}$
return $[\![(c^*, r \star \mathsf{pk}) \in \mathcal{T}]\!]$ $\quad\backslash\backslash\ \mathrm{G}_5$-$\mathrm{G}_{14}$
for $i \in [q_D]$ $\quad\backslash\backslash\ \mathrm{G}_7$
  $(a_i, z_i) \leftarrow \mathcal{S}.E(d_i)$ $\quad\backslash\backslash\ \mathrm{G}_7$

Oracle $\mathsf{Decaps}(\mathsf{sk}, (c, d))$ $\quad\backslash\backslash\ \mathrm{G}_1$-$\mathrm{G}_6$

if $(c, d) = (c^*, d^*)$ $\quad\backslash\backslash\ \mathrm{G}_1$
if $(d = d^*)$ $\quad\backslash\backslash\ \mathrm{G}_2$
if $(c = c^* \lor d = d^*)$ $\quad\backslash\backslash\ \mathrm{G}_3$-$\mathrm{G}_6$
  return $\perp$
if $\mathsf{G}(c, \mathsf{sk} \star c) \neq d$ $\quad\backslash\backslash\ \mathrm{G}_1$-$\mathrm{G}_5$
if $\mathcal{S}.RO(c, \mathsf{sk} \star c) \neq d$ $\quad\backslash\backslash\ \mathrm{G}_6$
  return $\perp$
return $\mathsf{H}(c, g \star c)$

Oracle $\textsc{Decaps}(\mathsf{sk}, (c, d))$ $\quad\backslash\backslash\ \mathrm{G}_7$-$\mathrm{G}_{12}$

if $(c = c^* \lor d = d^*)$ **return** $\perp$
$d' \leftarrow \mathcal{S}.RO(c, \mathsf{sk} \star c)$
$(a, z) \leftarrow \mathcal{S}.E(d)$ $\quad\backslash\backslash\ \mathrm{G}_8$-$\mathrm{G}_{12}$
if $(a, z) = \perp$ **return** $\perp$ $\quad\backslash\backslash\ \mathrm{G}_9$-$\mathrm{G}_{12}$
if $(a \neq c) \lor (z \neq \mathsf{sk} \star c)$ **return** $\perp$ $\quad\backslash\backslash\ \mathrm{G}_{10}$
if $(a \neq c) \lor (\mathrm{O}(a, z) = 0)$ **return** $\perp$ $\quad\backslash\backslash\ \mathrm{G}_{11}$-$\mathrm{G}_{12}$
if $d' \neq d$ **return** $\perp$
return $\mathsf{H}(a, z)$ $\quad\backslash\backslash\ \mathrm{G}_{12}$
return $\mathsf{H}(c, g \star c)$

Oracle $\textsc{Decaps}(\mathsf{sk}, (c, d))$ $\quad\backslash\backslash\ \mathrm{G}_{13}$-$\mathrm{G}_{14}$

if $(c = c^* \lor d = d^*)$ **return** $\perp$
$d' \leftarrow \mathcal{S}.RO(c, \mathsf{sk} \star c)$ $\quad\backslash\backslash\ \mathrm{G}_{13}$
$(a, z) \leftarrow \mathcal{S}.E(d)$
if $(a, z) = \perp \lor a \neq c$ **return** $\perp$
if $\mathrm{O}(a, z) = 0$ **return** $\perp$
return $\mathsf{H}(a, z)$

Fig. 14. Games G_1-G_{14} for the proof of Theorem 4. Ext is the extractor algorithm in the O2H lemma, which runs $\mathcal{A}$ and measures the input of one of the parallel random oracle queries uniformly at random.

$c \neq c^*$), since then G_2 return $\perp$ on elements where G_1 would return $\mathsf{H}(c, g \star c)$. By a straight-forward reduction to GDP we have

$$\left|\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1]\right| \leq 8(q+1)^2/2^n\,.$$

Game G_3. In this game we return $\perp$ on inputs (c^*, d). This will not be noticeable to the adversary, since the the hash check will not evaluate to true (since $d \neq d^*$) and so $\perp$ is returned in G_2 and in G_3. Thus, $\Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1]$.

Game G_4. Here we reprogram G and H on the challenge input using O2H (viewing them as a joint oracle $\mathsf{G} \times \mathsf{H}$). Note that due to the previous game-hops, we made sure that the random oracles will not be queried on challenge inputs in the decapsulation oracle. This is because Decaps returns $\perp$ if either the first input is c^* or the second input is d^*. Since Decaps does not query G or H on the reprogrammed challenge input, we do not need to consider indirect queries to G and H through $\textsc{Decaps}$ for the O2H lemma. Thus, we have

$$\left|\Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1]\right| \leq 2d\sqrt{\Pr[\mathrm{G}_5^{\mathsf{Ext}} \Rightarrow 1]}$$

and since b is now information-theoretically hidden in the view of the adversary

$$\Pr[\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1] = \tfrac{1}{2}\,,$$

where G_5 works as G_3 except that the input of the i-th parallel random oracle query, where $i \xleftarrow{\$} [d]$, is measured and 1 is returned if the challenge input was recovered, that is if if one of the p input registers contains $(c^*, r \star \mathsf{pk})$.

Game G_6. In this game we simulate random oracle G using the extractable random-oracle simulator from Definition 11. We set the commitment function f to be the output of G, that is we set $f(x, y) \mapsto y$. Since we do not yet make use of the extraction interface, we have by property 1 from Definition 11 that the extractable random-oracle simulator is *perfectly indistinguishable* from the quantum random oracle G. Therefore, this is only a conceptual change and we have $\Pr[\mathrm{G}_5^{\mathsf{Ext}} \Rightarrow 1] = \Pr[\mathrm{G}_6^{\mathsf{Ext}} \Rightarrow 1]$.

Game G_7. In G_7 we extract the ciphertexts which were queried on the decapsulation oracle *at the end of the game*, using the extraction interface $\mathcal{S}.E$. Since we only do this at the end of the game, it does not change the outcome of the game. Thus, this is again a conceptual game change and we have $\Pr[G_6^{\mathsf{Ext}} \Rightarrow 1] = \Pr[G_7^{\mathsf{Ext}} \Rightarrow 1]$.

Game G_8. We now extract the commitments *at runtime* in the decapsulation algorithm after querying $S.RO$. Since we have q_D ciphertexts which need to be extracted, and each one needs to be commuted at most $q + q_D$ times we have to apply the almost-commutativity property (Property 3 of Definition 11) at most $q_D(q + q_D)$ times. Observing that $\Gamma(f) = 1$ for $f(x, y) \mapsto y$ we have

$$\left|\Pr[G_7^{\mathsf{Ext}} \Rightarrow 1] - \Pr[G_8^{\mathsf{Ext}} \Rightarrow 1]\right| \le q_D(q_D + q)8\sqrt{\frac{1}{2^{n-1}}}.$$

Game G_9. We introduce the test whether the extracted $(a, z) = \perp$ and return $\perp$ if it evaluates to true. We can bound this using property 6 of Definition 11 together with a union bound over the number of decapsulation queries. Informally, property 6 says in our setting if $d = d'$, then $(a, z) \neq \perp$ except with probability $2^{-(n-1)}$ (for a single call). Thus, by the contrapositive this means that if $(a, z) = \perp$ then we have $d \neq d'$. By a union bound over the q_D decapsulation queries, we get

$$\left|\Pr[G_8^{\mathsf{Ext}} \Rightarrow 1] - \Pr[G_9^{\mathsf{Ext}} \Rightarrow 1]\right| \le \frac{q_D}{2^{n-1}}\,.$$

Game G_{10}. We introduce an additional test after the one introduced by G_9, in which we return $\perp$ if $(a, z) \neq \perp$ and $(a, z) \neq (c, \mathsf{sk} \star c)$. Intuitively, if $d = d'$ we have $(a, z) = (c, \mathsf{sk} \star c)$ unless the adversary has found a collision in the key-confirmation hash. If $d \neq d'$, we do not change the behavior in G_{10} because Decaps already returns $\perp$ as introduced in G_9. We apply property 8 of Definition 11 to bound the collision probability. We set q in property 8 to $q + q_D$ (the second q is the total number of RO queries) and $\ell = q_D$. The q_D term is necessary to account for indirect queries to G through Decaps. With the observation that $\Gamma'(f) = 1$ for our commitment function f, we obtain by applying property 8

$$\left|\Pr[G_9^{\mathsf{Ext}} \Rightarrow 1] - \Pr[G_{10}^{\mathsf{Ext}} \Rightarrow 1]\right| \le 40e^2(q + 2q_D + 2)^3/2^n\,.$$

Game G_{11}. We make a simple conceptual change, by substituting the previous check $z \neq \mathsf{sk} \star c$ with $O(a, z) = 0$, where we set O to be the function $\mathsf{GA\text{-}DDH}_g$. In the final reduction we simulate it using the (classical) $\mathsf{GA\text{-}DDH}_g$ oracle. We have $\Pr[G_{10}^{\mathsf{Ext}} \Rightarrow 1] = \Pr[G_{11}^{\mathsf{Ext}} \Rightarrow 1]$.

Game G_{12}. In this game we make another conceptual change. We return $\mathsf{H}(a, z)$ instead of $\mathsf{H}(c, g\star c)$ which is possible, since the previous condition ensures that if the line is reached, that $a = c$ and $O(a, z) = 1$ and thus $z = \mathsf{sk} \star c$. Therefore, $\Pr[G_{11}^{\mathsf{Ext}} \Rightarrow 1] = \Pr[G_{12}^{\mathsf{Ext}} \Rightarrow 1]$.

Game G_{13}. We now remove the returning of $\perp$ when $d \neq d'$. If $(a, z) \neq \perp$, we have by property 5 of Definition 11 that $\mathcal{S}.RO(\mathcal{S}.E(d)) = d$ except with probability $2^{-(n-1)}$. The previous check ensures that $a = c$ and $z = \mathsf{sk}\star c$. Thus, if the line is reached we have $d' = d$ due to property 5 since $\mathcal{S}.RO(c, \mathsf{sk}\star c) = d$. We can thus drop the check, as it will most likely evaluate to false. Thus, we have by a union bound over the q_D decapsulation queries

$$\left|\Pr[G_{12}^{\mathsf{Ext}} \Rightarrow 1] - \Pr[G_{13}^{\mathsf{Ext}} \Rightarrow 1]\right| \le \frac{q_D}{2^{n-1}}\,.$$

Game G_{14}. In G_{14} we move the queries $\mathcal{S}.RO(c, \mathsf{sk} \star c)$ to the end of the game, which is possible since we do not use their values. Observe that the secret key is not used in game G_{14} anymore, except for testing whether $O(a, z) = 0$, which the reduction will be able to simulate using the classical decision oracle. By the almost-commutativity, we have

$$\left|\Pr[G_{13}^{\mathsf{Ext}} \Rightarrow 1] - \Pr[G_{14}^{\mathsf{Ext}} \Rightarrow 1]\right| \le q_D(q_D + q)8\sqrt{\frac{1}{2^{n-1}}}.$$

It remains to bound $\Pr[G_{14}^{\mathsf{Ext}} \Rightarrow 1]$. Let $p \coloneqq q/d$ be the query parallelism. Since we can now simulate decapsulation queries using the decision oracle without using the secret key we can easily show that there exists an adversary $\mathcal{B}$ with

$$\Pr[G_{14}^{\mathsf{Ext}} \Rightarrow 1] \le \mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{GA\text{-}StCDH}}(\mathcal{B})\,,$$

where we use the fact that it is possible to “detect” the right solution in the set of measured queries due to the decision oracle. Detecting the right solution takes (at most) p queries to $\mathsf{GA\text{-}DDH}(g \star \tilde{x}, \cdot, \cdot)$. We describe how $\mathcal{B}$ works. It takes as input the challenge $(g \star \tilde{x}, h \star \tilde{x})$, then samples d^* and K_b as in G_{14}, sets c^* to be $h \star \tilde{x}$ and pk to be $g \star \tilde{x}$. It runs the extractor algorithm Ext as in G_{14}, it simulates G using the extractable random oracle simulator and simulates DECAPS queries as in G_{14} using the *classical* $\mathsf{GA\text{-}DDH}_g$ oracle. Finally, when it obtains $\mathcal{T}$, it searches for z s.t. $\mathsf{GA\text{-}DDH}_g(c^*, z) = 1$, and returns z. Clearly, when $(c^*, r \star \mathsf{pk}) \in \mathcal{T}$, $\mathcal{B}$ wins.

Adding up the terms yields the claimed bound. The claimed running time follows from running $\mathcal{A}$ once and Property 9 of the extractable random oracle simulator, which describes its running time depending on the number of hash and extraction queries, which concludes our proof. ☐

C.3 Proof of Theorem 6

Proof. We prove the theorem using the games in Figure 15.

Games G_1-G_5

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}$ $\quad\backslash\backslash\, G_1$-$G_2$

for $i \in [\ell - 1]$ $\quad\backslash\backslash\, G_3$-$G_5$

  $h_i \xleftarrow{\$} \mathcal{G}$

  $x_i := h_i \star \tilde{x}$

  $t_i := h_i$

for $i \in [\ell, m]$

  $\backslash\backslash$ Enforce $\{s_\ell, \ldots, s_{2\ell-1}\} = [0, \ell - 1]$

  $s_i \xleftarrow{\$} [0, \ell - 1]$

  $h_i \xleftarrow{\$} \mathcal{G}$

  $x_i := h_i \star x_{s_i}$ $\quad\backslash\backslash\, x_0 := \tilde{x}$

  $t_i := h_i \cdot h_{s_i}$ $\quad\backslash\backslash\, h_0 := e$

$\mathsf{pk} := (x_1, \ldots, x_m)$

$\mathsf{sk} := (t_1, \ldots, t_m)$

$b \xleftarrow{\$} \{0, 1\}$

$(\mathsf{ct}^*, K_0) \leftarrow \mathsf{Encaps}(\mathsf{pk})$ $\quad\backslash\backslash\, G_1$

$r \xleftarrow{\$} \mathcal{G}$ $\quad\backslash\backslash\, G_2$-$G_5$

$\mathsf{ct}^* := r \star \tilde{x}$

$K_0 := \mathsf{H}_1(\mathsf{ct}^*)$

$K_1 \xleftarrow{\$} \{0, 1\}^\kappa$

$b' \leftarrow \mathcal{A}^{\text{DECAPS}, \mathsf{H}}(\mathsf{pk}, c^*, K_b)$ $\quad\backslash\backslash\, G_1$-$G_4$

$b' \leftarrow \mathcal{A}^{\text{DECAPS}, \mathsf{H}'}(\mathsf{pk}, c^*, K_b)$ $\quad\backslash\backslash\, G_5$

return $[\![b = b']\!]$

Oracle $\mathsf{H}(M)/\mathsf{H}'(M)$

Let $M = (\mathsf{ct}, z_1, \ldots, z_m)$

if $z_i = \mathsf{sk}_i \star \mathsf{ct}$ for all $i \in [m]$ $\quad\backslash\backslash\, G_2$-$G_3$

  return $\mathsf{H}_1(\mathsf{ct})$

if $\mathsf{TDT}(\mathsf{ct}, z_1, \ldots, z_m) = 1$ $\quad\backslash\backslash\, G_4$-$G_5$

  return $\mathsf{H}_1(\mathsf{ct})$ $\quad\backslash\backslash\, \mathsf{H}$

  return $\mathsf{H}'_1(\mathsf{ct})$ $\quad\backslash\backslash\, \mathsf{H}'$

return $\mathsf{H}_2(M)$

Oracle $\text{DECAPS}(\mathsf{ct} \neq \mathsf{ct}^*)$

for $i \in [m]$ $\quad\backslash\backslash\, G_1$

  $z_i := \mathsf{sk}_i \star \mathsf{ct}$

return $\mathsf{H}(\mathsf{ct}, z_1, \ldots, z_m)$

return $\mathsf{H}_1(\mathsf{ct})$ $\quad\backslash\backslash\, G_2$-$G_5$

Fig. 15. Games G_1-G_5 for the proof of Theorem 6, H_1,H'_1 and H_2 are internal random oracles.

GAME G_1. This is the standard $\mathsf{IND\text{-}CCA}$ game for KEMs for $\mathsf{GA\text{-}Twin\text{-}HEG}_m$. The random oracle H is instantiated by an internal random oracle H_2. Thus,

$$\left|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \tfrac{1}{2}\right| = \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{GA\text{-}Twin\text{-}HEG}_m}(\mathcal{A}) \ .$$

GAME G_2. In G_2, we split the random oracle H into two internal random oracles H_1 and H_2. H_1 is used in the DECAPS oracle to compute the keys. We also make the conceptual change of only hashing the ciphertext ct without the z_i. This change is indeed only conceptual, since every possible set element is a valid encapsulation and all z_i are uniquely determined by ct and sk. So splitting it into a separate internal random oracle and only hashing ct does not change the distribution of the keys.

H now has to check whether a query corresponds to such a decapsulation query and in this case it answers with H_1 only on the first component of its query (which corresponds to ct in this case). In all other cases, it still hashes all its input with H_2. This makes H consistent with the decapsulation oracle, so overall, we have $\Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \Pr[G_2^{\mathcal{A}} \Rightarrow 1]$.

Game G_3. Next, we change how the challenge key is generated. Instead of choosing all set elements at random, we sample $\ell - 1$ "basis" set elements and set the remaining public key elements as rerandomizations of one of these bases, depending on a randomly chosen $s_i \xleftarrow{\$} [0, \ell - 1]$. For $s_i = 0$ we use the origin element $\tilde{x}$. Like in Lemma 3, we choose the s_i s.t. each base element is "used" once in the first ℓ rerandomized elements. This does not change the distribution of the public key (see Lemma 3), so $\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \Pr[G_3^{\mathcal{A}} \Rightarrow 1]$.

Game G_4. Finally, we replace the check whether a hash query corresponds to a decapsulation with the trapdoor test from Lemma 3. From the lemma, we get

$$|\Pr[G_3^{\mathcal{A}} \Rightarrow 1] - \Pr[G_4^{\mathcal{A}} \Rightarrow 1]| \leq \mathsf{Adv}^{\mathsf{TDT}}_{\mathsf{EGA},q,d,\ell,m}(\mathcal{D}_1) \leq \mathsf{Adv}^{\mathsf{GDP}}_{T,q,d}(\mathcal{D}_2) \ ,$$

for any quantum distinguishers $\mathcal{D}_1, \mathcal{D}_2$, where T defined as in Lemma 3 and q and d are the number of queries and the query depth of $\mathcal{A}$ to the random oracle H respectively.

Game G_5. Next, we replace the internal random oracle H_1 used in decapsulation and H with an internal random oracle H_1', which is identical to H_1 at all points except for ct^*. We call the new random oracle H'. We still use H_1 to compute the challenge key K_0, but $\mathcal{A}$ now only has access to H', so from the perspective of the adversary $\mathcal{A}$ both keys are now random. Therefore, we have

$$\Pr[G_5^{\mathcal{A}} \Rightarrow 1] = \tfrac{1}{2} \ .$$

It remains to bound the difference between G_4 and G_5. By the MRM O2H lemma, there exists an extractor algorithm Ext with

$$|\Pr[G_4^{\mathcal{A}} \Rightarrow 1] - \Pr[G_5^{\mathcal{A}} \Rightarrow 1]| \leq 4d \Pr[G_6^{\mathsf{Ext}} \Rightarrow 1] \ ,$$

where G_6^{Ext} is the same as G_4, except that instead of running $\mathcal{A}^{\textsc{Decaps},\mathsf{H}}(\mathsf{pk}, c^*, K_b)$ to obtain b' and returning $[\![b = b']\!]$, it runs $\mathsf{Ext}^{\textsc{Decaps},\mathsf{H},\mathsf{H}'}(\mathsf{pk}, c^*, K_b)$ to obtain $\mathcal{T}$ and returns $[\![\mathcal{S} \cap \mathcal{T} \neq \varnothing]\!]$, where $\mathcal{S}$ are the inputs of H which are reprogrammed, i.e., $\mathcal{S} = \{(\mathsf{ct}^*, r \star \mathsf{pk}_1, ..., r \star \mathsf{pk}_m)\}$.

We bound the right-hand probability by the adversary $\mathcal{B}$ given in Figure 16. $\mathcal{B}$ embeds its challenge x in the first element of the public key and uses its challenge y as the challenge encapsulation. It simulates internal random oracles H_1, H_1' and H_2 using standard techniques and runs the extraction algorithm Ext of the MRM O2H lemma to obtain a set $\mathcal{T}$. Note that the Decaps and H, H' can be simulated as in G_5 and quantum decapsulation queries can also be simulated since $\mathcal{B}$ has quantum access to H_1. Finally, once $\mathcal{T}$ is obtained, $\mathcal{B}$ searches for $(\mathsf{ct}^*, z_1, ..., z_m) \in \mathcal{T}$ and runs $\mathsf{TDT}(\mathsf{ct}^*, z_1, ..., z_m)$. If the trapdoor test passes, it returns z_1 as the solution. Since $\mathcal{B}$ wins, when Ext wins, we have

$$Pr[G_6^{\mathsf{Ext}} \Rightarrow 1] \leq \mathsf{Adv}^{\mathsf{GA\text{-}CDH}}_{\mathsf{EGA}}(\mathcal{B}) \ .$$

Collecting the probabilities and applying Lemma 2 yields the claim. □

D Omitted Proofs for Group Action NIKE Schemes

In order to make our NIKE proofs better understandable we capture the CKS security game described in Section 2.2 by the pseudocode given in Figure 17.

In Appendix D.2, we prove the necessity of the GA-DPQ-StCDH assumption. In Appendices D.3 and D.4, we then give the proofs for the security of GA-HDH based on the GA-DFQ-StCDH assumption and that of GA-Twin-HDH$_m$ based on the GA-CDH assumption. In Appendix D.1 we explain why it seems difficult to apply the MRM-O2H lemma to prove security in the CKS model.

D.1 Difficulty of Applying MRM to NIKE

As explained in Remark 8, in order to apply the MRM-O2H lemma, we have to show how to simulate the oracles in the CKS model quantumly in order to turn the adversary into a reversible one. Notice how the proof uses the precomputed set of public and secret-keys in the RegisterHonest oracle, which does not make sense for quantum queries since the same keys would be registered to different identities in the

Adversary $\mathcal{B}(x, y)$

$x_1 := x$
for $i \in [2, \ell - 1]$
  $h_i \xleftarrow{\$} \mathcal{G}$
  $x_i := h_i \star \tilde{x}$
for $i \in [\ell, m]$
  $\backslash\backslash$ Enforce $\{s_\ell, \ldots, s_{2\ell-1}\} = [0, \ell - 1]$
  $s_i \xleftarrow{\$} [0, \ell - 1]$
  $h_i \xleftarrow{\$} \mathcal{G}$
  $x_i := h_i \star x_{s_i}$ $\quad\backslash\backslash\, x_0 := \tilde{x}$
$\mathsf{pk} := (x_1, \ldots, x_m)$
$\mathsf{ct}^* := y$
$K_0 := \mathsf{H}_1(\mathsf{ct}^*)$
$K_1 \xleftarrow{\$} \{0,1\}^\kappa$
$\mathcal{T} \leftarrow \mathsf{Ext}^{\textsc{Decaps},\mathsf{H},\mathsf{H}'}(\mathsf{pk}, \mathsf{ct}^*, K_b)$
for $(\mathsf{ct}, z_1, \ldots, z_m) \in \mathcal{T}$ $\quad\backslash\backslash\, |\mathcal{T}| = p$
  if $\mathsf{ct} = \mathsf{ct}^* \wedge \mathsf{TDT}(\mathsf{ct}^*, z_1, \ldots, z_m) = 1$
    return z_1
return $\perp$

Fig. 16. Adversary $\mathcal{B}$ against $\mathsf{GA\text{-}CDH}$, where $\mathsf{H}, \mathsf{H}_1, \mathsf{H}_1'$ and H_2 are internal random oracles. The only constraint on the random oracles is that $\mathsf{H}_1(z) = \mathsf{H}_1'(z)$ for all $z \neq y$.

simulation. It is not clear how to generalize this technique without changing the CKS model to a slightly weaker one where the adversary first commits to the identities. Additionally, we would have to assume the existence of quantum-accesible RAM for the key registration set L and the test set K. Additionally, the id space would need to be sufficiently small for the reduction to have a somewhat reasonable memory usage. This additional freedom the adversary has in choosing the identities is not existent in the $\mathsf{IND\text{-}CCA}$ security game for KEMs, which is why it is not an issue there. We leave it as an open problem whether one can apply MRM-O2H also in the NIKE setting in the CKS model. For a more in-depth discussion regarding why a reversible adversary is necessary in order to apply MRM-O2H, see [27].

D.2 Proof of Theorem 2

Proof. Consider the sequence of games given in Figure 18.

Game 1. This is the $\mathsf{GA\text{-}DPQ\text{-}StCDH}$ game. By definition,

$$\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] = \mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{GA\text{-}DPQ\text{-}StCDH}}(\mathcal{A}).$$

Game 2. In this game, instead of returning whether $g \star x_1 = x_2$ in line 07, oracle O_g returns whether $(x_1, g \star x_1) = (x_1, x_2)$ in line 06. We do the same for oracle O_h (see line 08). Since we always have $x_1 = x_1$, this change is only conceptual and thus $\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1]$.

Game 3. In this game we want to change the output of O_g and O_h again. But first, we will prepare some conceptual changes which we will need in the final reduction to CKS security. Therefore, we add identities to the game and we introduce the notation $\mathsf{ID}_{\mathsf{int}(\sigma)} := \sigma$ for an identity string $\sigma \in \{0,1\}^\mu$, where int converts σ to an integer. This way, we can assign identities in ascending order using a counter cnt, starting with the challenge set elements $g \star \tilde{x}$ and $h \star \tilde{x}$ in line 02. After that we increment cnt for each query to O_g or O_h and assign new identities to all x_1 in these queries. Now we can change the output of O_g from deciding whether $(x_1, g \star x_1) = (x_1, x_2)$ in line 06 to deciding whether $\mathsf{H}(\mathsf{ID}_g, \mathsf{ID}, g \star \tilde{x}, x_1, g \star x_1) = \mathsf{H}(\mathsf{ID}_g, \mathsf{ID}, g \star \tilde{x}, x_1, x_2)$ (cf. lines 12, 14) using a hash function H. We do the same for O_h using h instead of g. This introduces false positives into the outputs, when for any $x_1 \in \mathcal{X}$ we have that $\mathsf{H}(\mathsf{ID}_g, \mathsf{ID}, g \star \tilde{x}, x_1, g \star x_1)$ has preimages of the form $(\mathsf{ID}_g, \mathsf{ID}, g \star \tilde{x}, x_1, x_2)$ with $x_2 \neq g \star x_1$.

We can bound this change by reducing to the Generic Distinguishing Problem, which we do in Figure 19. For every input $(\mathsf{ID}_g, \mathsf{ID}, g \star \tilde{x}, x_1, x_2)$ we have $\mathsf{F}(\mathsf{ID}_g, \mathsf{ID}, g \star \tilde{x}, x_1, x_2)$ returns 1 with probability $\lambda := 1/2^\kappa$, which is the probability to find a second preimage for $\mathsf{H}(\mathsf{ID}_g, \mathsf{ID}, g \star \tilde{x}, x_1, g \star x_1)$. The same holds for O_h. If F is the zero function, the distinguisher $\mathcal{D}$ simulates G_2 and otherwise G_3. Thus by

Game $\mathsf{Exp}^{\mathsf{CKS}}(\mathcal{A})$

00 $L[\mathcal{ID}] := \bot$
01 $K[\mathcal{ID}, \mathcal{ID}] := \bot$
02 $\mathsf{pp} \leftarrow \mathsf{NIKE.Setup}$
03 $b \xleftarrow{\$} \{0,1\}$
04 $b' \leftarrow \mathcal{A}^{\mathrm{O}}(\mathsf{pp})$
05 **return** $[\![b = b']\!]$

Oracle $\textsc{RegisterHonest}(\mathsf{ID})$

06 **if** $L[\mathsf{ID}] = (\mathit{corrupt}, *, \bot)$
07   **return** $\bot$
08 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{NIKE.Gen}(\mathsf{pp}, \mathsf{ID})$
09 $L[\mathsf{ID}] := (\mathit{honest}, \mathsf{pk}, \mathsf{sk})$
10 **return** $(\mathsf{ID}, \mathsf{pk})$

Oracle $\textsc{RegisterCorrupt}(\mathsf{ID}, \mathsf{pk})$

11 **if** $L[\mathsf{ID}] = \bot$
12    $L[\mathsf{ID}] := (\mathit{corrupt}, \mathsf{pk}, \bot)$
13   **return** 1
14 **else if** $L[\mathsf{ID}] = (\mathit{corrupt}, *, \bot)$
15    $L[\mathsf{ID}] := (\mathit{corrupt}, \mathsf{pk}, \bot)$
16   **return** 1
17 **return** 0

Oracle $\textsc{CorruptReveal}(\mathsf{ID}_1, \mathsf{ID}_2)$

18 **if** $L[\mathsf{ID}_1] = \bot \vee L[\mathsf{ID}_2] = \bot$
19   **return** $\bot$
20 $(l_1, \mathsf{pk}_1, \mathsf{sk}_1) := L[\mathsf{ID}_1],\ (l_2, \mathsf{pk}_2, \mathsf{sk}_2) := L[\mathsf{ID}_2]$
21 **if** $l_1 = l_2$
22   **return** $\bot$
23 Let $\mathsf{sk}_i \neq \bot$
24 **return** $\mathsf{NIKE.SharedKey}(\mathsf{ID}_{3-i}, \mathsf{pk}_{3-i}, \mathsf{ID}_i, \mathsf{sk}_i)$

Oracle $\textsc{Test}(\mathsf{ID}_1, \mathsf{ID}_2)$

25 **if** $L[\mathsf{ID}_1] = \bot \vee L[\mathsf{ID}_2] = \bot \vee \mathsf{ID}_1 = \mathsf{ID}_2$
26   **return** $\bot$
27 **if** $K[\mathsf{ID}_1, \mathsf{ID}_2] \neq \bot$
28   **return** $K[\mathsf{ID}_1, \mathsf{ID}_2]$
29 $(l_1, \mathsf{pk}_1, \mathsf{sk}_1) := L[\mathsf{ID}_1],\ (l_2, \mathsf{pk}_2, \mathsf{sk}_2) := L[\mathsf{ID}_2]$
30 **if** $l_1 \neq \mathit{honest} \vee l_2 \neq \mathit{honest}$
31   **return** $\bot$
32 $K_0 := \mathsf{NIKE.SharedKey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$
33 $K_1 \xleftarrow{\$} \mathcal{SHK}$
34 $K[\mathsf{ID}_1, \mathsf{ID}_2] := K_b,\ K[\mathsf{ID}_2, \mathsf{ID}_1] := K_b$
35 **return** K_b

Fig. 17. CKS security game $\mathsf{Exp}^{\mathsf{CKS}}$ for NIKE. Adversary $\mathcal{A}$ has access to oracles $\mathrm{O} = \{$RegisterHonest, RegisterCorrupt, CorruptReveal, Test$\}$.

Games G_1-G_4

00 $g \xleftarrow{\$} \mathcal{G}$
01 $h \xleftarrow{\$} \mathcal{G}$
02 $\mathsf{ID}_g := \mathsf{ID}_0;\ \mathsf{ID}_h := \mathsf{ID}_1$ $\backslash\!\backslash\ \mathrm{G}_3$-$\mathrm{G}_4$
03 $\mathsf{cnt} := 2$ $\backslash\!\backslash\ \mathrm{G}_3$-$\mathrm{G}_4$
04 $z \leftarrow \mathcal{A}^{\mathrm{O}_g(\cdot,|\cdot\rangle),\mathrm{O}_h(\cdot,|\cdot\rangle)}(g \star \tilde{x}, h \star \tilde{x})$
05 **return** $[\![z = gh \star \tilde{x}]\!]$

Oracle $\mathrm{O}_g(x_1, x_2)$ $\backslash\!\backslash\ \mathrm{G}_1$-$\mathrm{G}_2$

06 **return** $[\![(x_1, g \star x_1) = (x_1, x_2)]\!]$ $\backslash\!\backslash\ \mathrm{G}_2$
07 **return** $[\![g \star x_1 = x_2]\!]$

Oracle $\mathrm{O}_h(x_1, x_2)$ $\backslash\!\backslash\ \mathrm{G}_1$-$\mathrm{G}_2$

08 **return** $[\![(x_1, h \star x_1) = (x_1, x_2)]\!]$ $\backslash\!\backslash\ \mathrm{G}_2$
09 **return** $[\![h \star x_1 = x_2]\!]$

Oracle $\mathrm{O}_g(x_1, x_2)$ $\backslash\!\backslash\ \mathrm{G}_3$-$\mathrm{G}_4$

10 $\mathsf{ID} := \mathsf{ID}_{\mathsf{cnt}}$
11 $\mathsf{cnt} := \mathsf{cnt} + 1$
12 $K := \mathsf{H}(\mathsf{ID}_g, \mathsf{ID}, g \star \tilde{x}, x_1, g \star x_1)$ $\backslash\!\backslash\ \mathrm{G}_3$
13 $K := \mathsf{NIKE.SharedKey}(\mathsf{ID}, x_1, \mathsf{ID}_g, g)$ $\backslash\!\backslash\ \mathrm{G}_4$
14 **return** $[\![K = \mathsf{H}(\mathsf{ID}_g, \mathsf{ID}, g \star \tilde{x}, x_1, x_2)]\!]$

Oracle $\mathrm{O}_h(x_1, x_2)$ $\backslash\!\backslash\ \mathrm{G}_3$-$\mathrm{G}_4$

15 $\mathsf{ID} := \mathsf{ID}_{\mathsf{cnt}}$
16 $\mathsf{cnt} := \mathsf{cnt} + 1$
17 $K := \mathsf{H}(\mathsf{ID}_h, \mathsf{ID}, h \star \tilde{x}, x_1, h \star x_1)$ $\backslash\!\backslash\ \mathrm{G}_3$
18 $K := \mathsf{NIKE.SharedKey}(\mathsf{ID}, x_1, \mathsf{ID}_h, h)$ $\backslash\!\backslash\ \mathrm{G}_4$
19 **return** $[\![K = \mathsf{H}(\mathsf{ID}_h, \mathsf{ID}, h \star \tilde{x}, x_1, x_2)]\!]$

Fig. 18. Games G_1-G_4. For games G_3-G_4, we use the implicit notation $\mathsf{ID}_{\mathsf{int}(\sigma)} := \sigma$ for an identity string $\sigma \in \{0,1\}^{\mu}$, where int converts σ to an integer.

eq. (2) of Lemma 2 where we have set $\lambda := 1/2^{\kappa}$ we have

$$\left|\Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1]\right| = \left|\Pr[\mathsf{GDP}_{\mathsf{F},0}^{\mathcal{D}} \Rightarrow 1] - \Pr[\mathsf{GDP}_{\mathsf{F},1}^{\mathcal{D}} \Rightarrow 1]\right| \leq 8(q+1)^2/2^{\kappa}\,.$$

Game 4. In this game we substitute the boolean test in O_g and O_h again. We now check whether $\mathsf{NIKE.SharedKey}(\mathsf{ID}, x_1, \mathsf{ID}_g, g) = \mathsf{H}(\mathsf{ID}_g, \mathsf{ID}, g \star \tilde{x}, x_1, x_2)$ in line 13 in O_g and the same for h in O_h. By definition of the shared key algorithm of $\mathsf{GA\text{-}HDH}$ this change is again only conceptual. Note that by assigning identities in an ascending order, we always have $\mathsf{ID}_g < \mathsf{ID}_h < \mathsf{ID}$ for each ID chosen in O_g or O_h which fixes the order of inputs to H. Hence, $\Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1]$.
It remains to bound G_4. We claim

$$\Pr[\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1] \leq 2 \cdot \mathsf{Adv}_{\mathsf{GA\text{-}HDH}}^{\mathsf{CKS}}(\mathcal{B}) + \tfrac{1}{2^{\kappa}}\,. \tag{12}$$

We construct an adversary $\mathcal{B}$ in Figure 20. $\mathcal{B}$ has access to oracles RegisterHonest, RegisterCorrupt, CorruptReveal, Test provided by the CKS game as well as random oracle H. It simulates G_4 as follows.

Distinguisher $\mathcal{D}^{\mathsf{F}}$	**Oracle** $\mathrm{O}_g(x_1, x_2)$
$g \xleftarrow{\$} \mathcal{G}$	**if** $g \star x_1 = x_2$ **return** 1
$h \xleftarrow{\$} \mathcal{G}$	$\mathsf{ID} := \mathsf{ID}_{\mathsf{cnt}}$
$\mathsf{ID}_g := \mathsf{ID}_0$; $\mathsf{ID}_h := \mathsf{ID}_1$	$\mathsf{cnt} := \mathsf{cnt} + 1$
$\mathsf{cnt} := 2$	**return** $\mathsf{F}(\mathsf{ID}_g, \mathsf{ID}, g \star \tilde{x}, x_1, x_2)$
$z \leftarrow \mathcal{A}^{\mathrm{O}_g(\cdot,\vert\cdot\rangle),\mathrm{O}_h(\cdot,\vert\cdot\rangle)}(g \star \tilde{x}, h \star \tilde{x})$	
return $[\![z = gh \star \tilde{x}]\!]$	**Oracle** $\mathrm{O}_h(x_1, x_2)$
	if $h \star x_1 = x_2$ **return** 1
	$\mathsf{ID} := \mathsf{ID}_{\mathsf{cnt}}$
	$\mathsf{cnt} := \mathsf{cnt} + 1$
	return $\mathsf{F}(\mathsf{ID}_h, \mathsf{ID}, h \star \tilde{x}, x_1, x_2)$

Fig. 19. Distinguisher $\mathcal{D}$ for the Generic Distinguishing Problem to bound G_2-G_3.

Adversary $\mathcal{B}^{\mathrm{O}_{\mathsf{CKS}}}$	**Oracle** $\mathrm{O}_g(x_1, x_2)$
$\mathsf{ID}_g := \mathsf{ID}_0$; $\mathsf{ID}_h := \mathsf{ID}_1$	$\mathsf{ID} := \mathsf{ID}_{\mathsf{cnt}}$
$\mathsf{cnt} := 2$	$\mathsf{cnt} := \mathsf{cnt} + 1$
$\mathsf{pk}_0 \leftarrow \textsc{RegisterHonest}(\mathsf{ID}_g)$	$\textsc{RegisterCorrupt}(\mathsf{ID}, x_1)$
$\mathsf{pk}_1 \leftarrow \textsc{RegisterHonest}(\mathsf{ID}_h)$	**return** $[\![\textsc{CorruptReveal}(\mathsf{ID}_0, \mathsf{ID}) = \mathsf{H}(\mathsf{ID}_0, \mathsf{ID}, \mathsf{pk}_0, x_1, x_2)]\!]$
$K \leftarrow \textsc{Test}(\mathsf{ID}_g, \mathsf{ID}_h)$	
$z \leftarrow \mathcal{A}^{\mathrm{O}_g(\cdot,\vert\cdot\rangle),\mathrm{O}_h(\cdot,\vert\cdot\rangle)}(\mathsf{pk}_0, \mathsf{pk}_1)$	**Oracle** $\mathrm{O}_h(x_1, x_2)$
return $[\![K \neq \mathsf{H}(\mathsf{ID}_0, \mathsf{ID}_1, \mathsf{pk}_0, \mathsf{pk}_1, z)]\!]$	$\mathsf{ID} := \mathsf{ID}_{\mathsf{cnt}}$
	$\mathsf{cnt} := \mathsf{cnt} + 1$
	$\textsc{RegisterCorrupt}(\mathsf{ID}, x_1)$
	return $[\![\textsc{CorruptReveal}(\mathsf{ID}_1, \mathsf{ID}) = \mathsf{H}(\mathsf{ID}_1, \mathsf{ID}, \mathsf{pk}_1, x_1, x_2)]\!]$

Fig. 20. Adversary $\mathcal{B}$ for bounding G_4, where $\mathrm{O}_{\mathsf{CKS}} = \{\textsc{RegisterHonest}, \textsc{RegisterCorrupt}, \textsc{CorruptReveal}, \textsc{Test}, \mathsf{H}\}$ and $\mathsf{ID}_{\mathsf{int}(\sigma)} := \sigma$ for an identity string $\sigma \in \{0,1\}^{\mu}$.

It first creates two honest users with IDs ID_0 and ID_1 using the RegisterHonest oracle. It receives the corresponding public keys pk_0 and pk_1 and directly issues a test query on these two users. Thus, it will receive either their shared key or a random key. We denote this challenge key by K. Now $\mathcal{B}$ runs adversary $\mathcal{A}$ against GA-DPQ-StCDH on input $(\mathsf{pk}_0, \mathsf{pk}_1)$. Hence, oracle O_g is defined with respect to pk_0 and O_h to pk_1. On each query (x_1, x_2) to one of these oracles, $\mathcal{B}$ chooses a new identity ID by incrementing the counter and registers a dishonest party with public key x_1. Recall that this input is classical. Then it reveals the key between this freshly generated user and ID_0 or ID_1 (depending on the oracle) and compares the result with $\mathsf{H}(\mathsf{ID}_0, \mathsf{ID}, \mathsf{pk}_0, x_1, x_2)$ or $\mathsf{H}(\mathsf{ID}_1, \mathsf{ID}, \mathsf{pk}_1, x_1, x_2)$ respectively, which will be the oracle's output.

Finally, $\mathcal{A}$ will output a solution z and $\mathcal{B}$ checks whether $\mathsf{H}(\mathsf{ID}_0, \mathsf{ID}_1, \mathsf{pk}_0, \mathsf{pk}_1, z)$ equals the challenge key. If this is the case, it returns $b' = 0$ (real), otherwise it returns $b' = 1$ (random). Clearly, if z is a solution to the computational group action problem of pk_0 and pk_1, then $\mathcal{B}$ always wins the CKS game in the real world. In the random world, it will win only with probability $1 - 1/2^{\kappa}$ since the challenge key might be the same as the real key with probability $1/2^{\kappa}$. When z is not the correct solution and K is the real key, then $\mathcal{B}$ will only win if the output of H still coincides with K, i.e. with probability $1/2^{\kappa}$. However, if K is a random key, $\mathcal{B}$ will win again with probability $1 - 1/2^{\kappa}$. Collecting the conditional probabilities yields the bound claimed in eq. (12), which also concludes our proof. □

D.3 Proof of Theorem 5

Proof. Let $\mathcal{A}$ be a quantum adversary as in Theorem 5. Consider the sequence of games given in Figure 21. We proceed by analyzing the different games.

Game G_1. This the original CKS game with the definition of the NIKE unrolled, except that we do not explicitly do the canonical reordering of the IDs and public keys in order to focus on the important parts of the proof. By definition,

$$\left|\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] - 1/2\right| = \mathsf{Adv}^{\mathsf{CKS}}_{\mathsf{GA\text{-}HDH}}(\mathcal{A}) .$$

Game G_2. In this game we make a conceptual change by precomputing the set of honest keys and storing them in a set $\mathcal{H}$. Since this does not change the distribution of the generated keys, this does

Games G_1-G_5

$L[\mathcal{ID}] := \perp$
$K[\mathcal{ID}, \mathcal{ID}] := \perp$
$\mathsf{pp} \leftarrow \mathsf{NIKE.Setup}$
$\mathsf{ctr} := 0$ $\quad /\!\!/ \mathrm{G}_2$-$\mathrm{G}_5$
$\mathcal{H} := \perp$
for $i \in [q_O]$
$\quad \mathcal{H}[i] := (\mathsf{pk}, \mathsf{sk}) := (g \star \tilde{x}, g \xleftarrow{\$} \mathcal{G})$
$b \xleftarrow{\$} \{0, 1\}$
$b' \leftarrow \mathcal{A}^{\mathsf{O},\mathsf{H}}(\mathsf{pp})$
return $[\![b = b']\!]$

Oracle $\textsc{Test}(\mathsf{ID}_1, \mathsf{ID}_2)$

if $L[\mathsf{ID}_1] = \perp \lor L[\mathsf{ID}_2] = \perp \lor \mathsf{ID}_1 = \mathsf{ID}_2$
$\quad$ **return** $\perp$
if $K[\mathsf{ID}_1, \mathsf{ID}_2] \neq \perp$
$\quad$ **return** $K[\mathsf{ID}_1, \mathsf{ID}_2]$
$(l_1, \mathsf{pk}_1, \mathsf{sk}_1) := L[\mathsf{ID}_1]$, $(l_2, \mathsf{pk}_2, \mathsf{sk}_2) := L[\mathsf{ID}_2]$
if $l_1 \neq honest \lor l_2 \neq honest$
$\quad$ **return** $\perp$
$K_0 := \mathsf{H}(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{sk}_2 \star \mathsf{pk}_1)$
$K_0 := \mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ $\quad /\!\!/ \mathrm{G}_4$-$\mathrm{G}_5$
$K_1 \xleftarrow{\$} \{0, 1\}^\kappa$
$K[\mathsf{ID}_1, \mathsf{ID}_2] := K_b$, $K[\mathsf{ID}_2, \mathsf{ID}_1] := K_b$
return K_b

Oracle $\textsc{RegisterHonest}(\mathsf{ID})$

if $L[\mathsf{ID}] = (corrupt, *, \perp)$ **return** $\perp$
$(\mathsf{pk}, \mathsf{sk}) := \mathcal{H}[\mathsf{ctr}]$ $\quad /\!\!/ \mathrm{G}_2$-$\mathrm{G}_5$
$\mathsf{ctr} := \mathsf{ctr} + 1$
$(\mathsf{pk}, \mathsf{sk}) := (g \star \tilde{x}, g \xleftarrow{\$} \mathcal{G})$ $\quad /\!\!/ \mathrm{G}_1$
$L[\mathsf{ID}] := (honest, \mathsf{pk}, \mathsf{sk})$
return $(\mathsf{ID}, \mathsf{pk})$

Oracle $\textsc{CorruptReveal}(\mathsf{ID}_1, \mathsf{ID}_2)$

if $(L[\mathsf{ID}_1] = \perp \lor L[\mathsf{ID}_2] = \perp)$ **return** $\perp$
$(l_1, \mathsf{pk}_1, \mathsf{sk}_1) := L[\mathsf{ID}_1]$, $(l_2, \mathsf{pk}_2, \mathsf{sk}_2) := L[\mathsf{ID}_2]$
if $l_1 = l_2$ **return** $\perp$
Let $\mathsf{sk}_i \neq \perp$
Let $\mathsf{pk}_i \in \mathcal{H}$ $\quad /\!\!/ \mathrm{G}_4$-$\mathrm{G}_5$
if $(\mathsf{pk}_{3-i}, *) \notin \mathcal{H}$
$\quad$ **return** $\mathsf{H}_1(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$
if $(\mathsf{pk}_{3-i}, *) \in \mathcal{H}$
$\quad$ **return** $\mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$
return $\mathsf{H}(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{sk}_i \star \mathsf{pk}_{3-i})$

Oracle $\mathsf{H}(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z)$ $\quad /\!\!/ \mathrm{G}_3$-$\mathrm{G}_5$

for $i \in [1, 2]$
$\quad$ **if** $(\mathsf{pk}_i, *) \in \mathcal{H} \land (\mathsf{pk}_{3-i}, *) \notin \mathcal{H}$
$\quad\quad$ Let sk_i s.t. $(\mathsf{pk}_i, \mathsf{sk}_i) \in \mathcal{H}$
$\quad\quad$ **if** $\mathsf{sk}_i \star \mathsf{pk}_{3-i} = z$
$\quad\quad\quad$ **return** $\mathsf{H}_1(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$
if $(\mathsf{pk}_1, *), (\mathsf{pk}_2, *) \in \mathcal{H}$
$\quad$ Let sk_1 s.t. $(\mathsf{pk}_1, \mathsf{sk}_1) \in \mathcal{H}$
$\quad$ **if** $\mathsf{sk}_1 \star \mathsf{pk}_2 = z$
$\quad\quad$ **return** $\mathsf{H}'_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ $\quad /\!\!/ \mathrm{G}_5$
$\quad\quad$ **return** $\mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ $\quad /\!\!/ \mathrm{G}_3$-$\mathrm{G}_4$
return $\mathsf{H}_2(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z)$

Fig. 21. Games G_1-G_5 for the proof of Theorem 5. The adversary $\mathcal{A}$ has access to oracles $\mathsf{O} = \{\textsc{RegisterHonest}, \textsc{RegisterCorrupt}, \textsc{CorruptReveal}, \textsc{Test}\}$ and H. $\textsc{RegisterCorrupt}$ is defined as in the original game in Figure 17. For improved readability we assume wlog $\mathsf{ID}_1 < \mathsf{ID}_2$. H_i for $i \in [1, 3]$ and H'_3 are internal random oracles and we define $\mathsf{H} := \mathsf{H}_2$ for G_1-G_2.

not change the distribution of $\textsc{RegisterHonest}$. Therefore, this change is only conceptual and we have $\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1]$.

$\textsc{Game}$ G_3 In this game, we make the following conceptual changes in H. If one of the public keys pk_i, where $i \in \{1, 2\}$, is from the honest set $\mathcal{H}$, and the other key pk_{3-i} *is not*, we check whether z is a valid DH tuple, i.e., $z = \mathsf{sk}_i \star \mathsf{pk}_{3-i}$, where sk_i is the corresponding secret key to pk_i stored in $\mathcal{H}$. If this test evaluates to true we return $\mathsf{H}_1(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ for an *internal* independent random oracle H_1. If *both keys* are from $\mathcal{H}$ and $z = \mathsf{sk}_1 \star \mathsf{pk}_2$, we return $\mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$, for another internal, independent random oracle H_3. In all other cases we return $\mathsf{H}_2(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z)$ for another independent internal random oracle H_2.

Looking ahead to the final reduction, these changes will allow us later to simulate the $\textsc{CorruptReveal}$ queries without the secret key because for all checks we can use the $\mathsf{GA\text{-}DDH}$ oracle. Additionally, it will allow us to properly see which values of the random oracle H we need to reprogram, namely the ones which can be forwarded to H_3. Since all these changes are only conceptual (see also the proof of Theorem 3 for a more detailed explanation, Section 4.1) we have $\Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1]$.

$\textsc{Game}$ G_4. Next, in the $\textsc{Test}$ oracle we now derive K_0 using H_3, which is only conceptual, by our new definition of H. Additionally, we return $\mathsf{H}_1(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ for $\textsc{CorruptReveal}$ queries if only one of the keys lies in $\mathcal{H}$ and $\mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ if both keys are in $\mathcal{H}$. Since $\textsc{CorruptReveal}$ previously always computed z correctly, its queries to H were forwarded to either H_1 or H_3 in G_3 as well. So by replacing the call to H with the same call to H_1 or H_3 that H would make is only a conceptual change and we have $\Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1]$.

Game G_5. In this game we reprogram the random oracle H on *all potential test queries* by reprogramming H on the set $\mathcal{S} := \{(*, *, \mathsf{pk}_1, \mathsf{pk}_2, \mathsf{sk}_1 \star \mathsf{pk}_2) \mid \text{for } i, j \in [q_O] \colon (\mathsf{pk}_1, \mathsf{sk}_1) = \mathcal{H}[i], (\mathsf{pk}_2, *) = \mathcal{H}[j] \text{ and } i \neq j\}$ with uniformly random values from $\{0,1\}^\kappa$. Note that this reprogramming does not affect inputs which get forwarded to H_1 by H, since the reprogramming happens on values where both keys are in the honest set, which are forwarded to H_3. Therefore, we derive the reprogrammed values using another internal random oracle called H_3'. Let H' be the reprogrammed random oracle, as described above. Since the queries to H_3 in CorruptReveal have different IDs then the ones from the Test queries, we have have that K_0 is now identically distributed to K_1 from the view of $\mathcal{A}$. Therefore, b is independent of $\mathcal{A}$'s view and we have

$$\Pr[G_5^{\mathcal{A}} \Rightarrow 1] = \tfrac{1}{2}\,.$$

It remains to bound G_4-G_5. By the semi-classical O2H lemma, the difference between the games is bounded by the *classical event* Find when running $\mathcal{A}$ with the punctured oracle $\mathsf{H} \setminus \mathcal{S}$. The event Find is set to 1 if the semi-classical oracle measures that an input x lies in $\mathcal{S}$. The main difficulty when applying the lemma is to show how to simulate the membership testing $x \in \mathcal{S}$. We solve this issue using the quantum-accessible double-sided decision oracle, in conjunction with the precomputed set of key pairs $\mathcal{H}$. Let $G_6^{\mathcal{A}}$ be as $G_4^{\mathcal{A}}$ except that $\mathcal{A}$ gets access to $\mathsf{H} \setminus \mathcal{S}$, we have

$$\left|\Pr[G_4^{\mathcal{A}} \Rightarrow 1] - \Pr[G_5^{\mathcal{A}} \Rightarrow]\right| \leq 2\sqrt{(d+1)\Pr[\textsc{Find}\colon G_6^{\mathcal{A}}]}\,.$$

We bound the right-hand probability by the adversary $\mathcal{B}$ given in Figure 22. $\mathcal{B}$ obtains the challenge (x_0, x_1) and the two corresponding decision oracles O_0, O_1. It embeds its challenges into each challenge public key pk by choosing a random bit $c \in \{0,1\}$ and group element $h \in \mathcal{G}$ and rerandomizing its challenge x_c with h, i.e. $\mathsf{pk} := h \star x_c$, where h and c are chosen separately and randomly for each public key. It simulates internal random oracles H_1, H_2, H_3 and H_3' using standard techniques and runs $\mathcal{A}^{O, \mathsf{H} \setminus \mathcal{S}}$ until the event $\textsc{Find} = 1$, then measures the corresponding input registers of $\mathsf{H} \setminus \mathcal{S}$ to obtain a set $\mathcal{T}$. It simulates all the oracles as in G_4, which $\mathcal{B}$ can do, since for all checks where secret keys are involved it can query the corresponding decision oracle. The punctured random oracle $\mathsf{H} \setminus \mathcal{S}$ is simulated as described in Figure 22, where the only difference to G_4 is that $\mathcal{B}$ takes account for the rerandomization factor h. Finally, once $\mathcal{T}$ is obtained, it searches for $(*, *, \mathsf{pk}_1, \mathsf{pk}_2, z) \in \mathcal{T}$, where pk_1 was derived by x_c using h_1 and pk_2 was derived by x_{1-c} using h_2 and $O_c(\mathsf{pk}_2, h_2^{-1} \star z) = 1$. If this check passes, it returns $(h_2^{-1} \cdot h_1^{-1}) \star z$ as the solution. First note that the bits c are information-theoretically hidden from the adversary. Thus, a test query will involve two public keys with different bits c with probability 1/2. Assuming that this is the case, $\mathcal{B}$ wins when $\textsc{Find} = 1$. This yields

$$\Pr[\textsc{Find}\colon G_6^{\mathcal{A}}] \leq 2\mathsf{Adv}_{\mathsf{EGA}}^{\mathsf{GA\text{-}DFQ\text{-}StCDH}}(\mathcal{B})\,.$$

Adding up the analyzed bounds yields the claimed bound. We conclude our proof by analyzing the running time of $\mathcal{B}$. The first part of $\mathcal{B}$ is the rerandomization of the keys, which is proportional to q_O assuming constant running time for a single rerandomization. Then there is the overhead for simulating the internal random oracles for running $\mathcal{A}$. For a single run of $\mathcal{A}$, H_1 is called at most $\max\{q, q_R\}$ times, H_2 is called at most q times, H_3 and H_3' are called at most $\max\{q, q_T\}$ times. Extracting the solution then is roughly proportional to p calls to the decision oracle. Every call H calls the decision oracles at most 2 times and and every call to the semi-classical oracle calls the decision oracle at most 2 times (a second call is made when there is another pk_1 but derived from the other set element), thus we have about at most $p + 4q$ calls to the decision oracles. Adding up and applying $\mathcal{O}$ notation yields the claimed running time, which concludes our proof.

□

D.4 Proof of Theorem 7

Proof. We prove the theorem using the games G_1-G_7 defined in Figure 23.

Game G_1. This is the standard CKS security game, so

$$\mathsf{Adv}_{\mathsf{GA\text{-}Twin\text{-}HDH}_m}^{\mathsf{CKS}}(\mathcal{A}) = |\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - 1/2|\,.$$

Adversary $\mathcal{B}^{\mathrm{O}_0(|\cdot\rangle,|\cdot\rangle),\mathrm{O}_1(|\cdot\rangle,|\cdot\rangle)}(x_0, x_1)$

$L[\mathcal{ID}] := \bot$
$K[\mathcal{ID}, \mathcal{ID}] := \bot$
$\mathsf{pp} \leftarrow \mathsf{NIKE.Setup}$
$\mathsf{ctr} := 0$
$\mathcal{H} := \bot$
for $i \in [q_O]$
  $c_i \xleftarrow{\$} \{0,1\}$
  $h_i \xleftarrow{\$} \mathcal{G}$
  $\mathcal{H}[i] := (\mathsf{pk}, h_i, c_i) := (r_i \star x_{c_i}, h_i, c_i)$
$b \xleftarrow{\$} \{0,1\}$
$\mathcal{T} \leftarrow$ Run $\mathcal{A}^{\mathrm{O},\mathsf{H}\setminus\mathcal{S}}(\mathsf{pp})$ until FIND
  and measure query register inputs
return $\mathsf{FindSolution}(\mathcal{T}, \mathcal{H})$

Oracle $\mathrm{O}_{\mathcal{S}}^{\mathsf{SC}}(|\psi_{in}, 0\rangle)$

$b := 0$
Parse ψ_{in} as $(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z)$
$b \leftarrow$ Measure [
  $\exists i, j \in [q_O] \wedge i \neq j, (*, h, c) \in \mathcal{H}$:
  $[\![(\mathsf{pk}_1, h, c) = \mathcal{H}[i] \wedge (\mathsf{pk}_2, *) = \mathcal{H}[j]]\!]$
  $\wedge\ [\![\mathrm{O}_c(\mathsf{pk}_2, h^{-1} \star z) = 1]\!]]$
return $(|\psi'_{in}\rangle, b)$

Oracle $\mathsf{H} \setminus \mathcal{S}(|\psi_{in}, \psi_{out}\rangle)$

$|\psi'_{in}, b\rangle \leftarrow \mathrm{O}_{\mathcal{S}}^{\mathsf{SC}}(|\psi_{in}, 0\rangle)$
if $b = 1$
  FIND $:= 1$
return $U_{\mathsf{H}}(|\psi'_{in}, \psi_{out}\rangle)$

Oracle $\mathsf{H}(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z)$

for $i \in [1, 2]$
  if $(\mathsf{pk}_i, r, c) \in \mathcal{H} \wedge (\mathsf{pk}_{3-i}, *, *) \notin \mathcal{H}$
  $\wedge\ \mathrm{O}_c(\mathsf{pk}_{3-i}, h^{-1} \star z) = 1$
    return $\mathsf{H}_1(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$
if $(\mathsf{pk}_1, h, c), (\mathsf{pk}_2, *, *) \in \mathcal{H} \wedge \mathrm{O}_c(\mathsf{pk}_2, h^{-1} \star z) = 1$
  return $\mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$
return $\mathsf{H}_2(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z)$

Function $\mathsf{FindSolution}(\mathcal{T}, \mathcal{H})$

$\mathcal{H}_0 := \{(a, b, 0) \in \mathcal{H}\}$
$\mathcal{H}_1 := \{(a, b, 1) \in \mathcal{H}\}$
for $(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z) \in \mathcal{T}$
  if $\exists h_1, h_2 : (\mathsf{pk}_1, h_1, c) \in \mathcal{H} \wedge (\mathsf{pk}_2, h_2, 1 - c) \in \mathcal{H}$
    if $\mathrm{O}_c(\mathsf{pk}_2, h_2^{-1} \star z)$
      return $(h_1^{-1} \cdot h_2^{-1}) \star z$
return $\bot$

Fig. 22. Adversary $\mathcal{B}$ for the proof of Theorem 5. All oracles (except for the random oracles) are simulated as in G_5, where H_1,H_2,H_3 and H'_3 are internal random oracles. The oracle $\mathrm{O}_{0/1}(|\cdot\rangle, |\cdot\rangle)$ denotes $\mathsf{GA\text{-}DDH}(g/h, |\cdot\rangle, |\cdot\rangle)$. We denote by U_{H} the unitary which implements the evaluation of H.

GAME G_2. In this game we make a conceptual change by precomputing the set of honest keys. We use a set $\mathcal{H}$ to store these key pairs. Since this does not change the distribution of the generated keys, this does not change the distribution of REGISTERHONEST. Therefore, this change is only conceptual and we have

$$\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1].$$

GAME G_3. In this game, we make the following conceptual changes in H. If one of the public keys pk_i, where $i \in \{1, 2\}$, is from the honest set $\mathcal{H}$, and the other key pk_{3-i} *is not*, we check whether all $z_{j,k}$ are correct. Wlog we assume the honest key is pk_1. Then we check $z_{j,k}$ by computing $z'_{j,k} = \mathsf{sk}_1[k] \star \mathsf{pk}_2[j]$ and comparing all values. If this test evaluates to true we return $\mathsf{H}_1(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ for an *internal* independent random oracle H_1. If *both keys* are from $\mathcal{H}$ and all $z_{j,k} = \mathsf{sk}_1[k] \star \mathsf{pk}_2[j]$, we return $\mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$, for another internal, independent random oracle H_3. In all other cases we return $\mathsf{H}_2(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z)$ for another independent internal random oracle H_2. Note that the CORRUPTREVEAL oracle still queries H, so it cannot be used to detect this change.

Looking ahead to the final reduction, these changes will allow us later to simulate the CORRUPTREVEAL queries without the secret key because for all checks we will use the trapdoor test, which we will introduce in the next game. Additionally, it will allow us to properly see which values of the random oracle H we need to reprogram, namely the ones which can be forwarded to H_3. Since all these changes are only conceptual, we have $\Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1]$.

GAME G_4. In this game we return $\mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ for TEST queries and $\mathsf{H}_1(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ for CORRUPTREVEAL queries if one key is not in the honest set and $\mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ if both keys are from the honest set. The latter can occur if the adversary reregisters an honest key as a malicious key with a different ID. For both oracles, the same queries are still answered with the same internal random oracle. Therefore, this is only a conceptual change and we have $\Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_4^{\mathcal{A}} \Rightarrow 1]$.

GAME G_5. In this game, we change how the honest keys are generated in preparation for the trapdoor test. Specifically, we choose 2 "base" set elements y_0 and y_1 and flip a coin for every key whether we use y_0 or y_1. The first public key element of each party is a rerandomization of the respective base set element. The next public key elements until the $(\ell - 1)$-th element will be computed as in the original protocol using the origin element $\tilde{x}$. The remaining $m - \ell$ elements are computed using one of the first $\ell - 1$ previously chosen public key elements. Therefore, we draw an index s_k randomly from $[0, \ell - 1]$ and take the s_k-th public key element for rerandomization. For $s_k = 0$, we simply use $\tilde{x}$. We now additionally

Games G_1-G_7

$L[\mathcal{ID}] := \bot$
$K[\mathcal{ID}, \mathcal{ID}] := \bot$
$\mathsf{pp} \leftarrow \mathsf{NIKE.Setup}$
$\mathsf{ctr} := 0$
$\mathcal{H} := \bot$
for $i \in [q_O]$ $\quad\backslash\backslash\, G_2$-$G_4$
$\quad (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{NIKE.Gen}(\mathsf{pp})$
$\quad \mathcal{H}[i] := (\mathsf{pk}, \mathsf{sk}, \bot)$
$r_0, r_1 \xleftarrow{\$} \mathcal{G}$ $\quad\backslash\backslash\, G_5$-$G_7$
$y_i := r_i \star \tilde{x}$ for $i \in \{0, 1\}$
for $i \in [q_O]$
$\quad c \xleftarrow{\$} \{0, 1\}$
$\quad$ **for** $k \in [m]$
$\quad\quad h_k \xleftarrow{\$} \mathcal{G}$
$\quad x_1 := h_1 \star y_c$
$\quad t_1 := h_1 \cdot r_c$
$\quad$ **for** $k \in [2, \ell - 1]$
$\quad\quad x_k := h_k \star \tilde{x}$
$\quad\quad t_k := h_k$
$\quad$ **for** $k \in [\ell, m]$
$\quad\quad \backslash\backslash$ Enforce $\{s_\ell, \ldots, s_{2\ell-1}\} = [0, \ell - 1]$
$\quad\quad s_k \xleftarrow{\$} [0, \ell - 1]$
$\quad\quad x_k := h_k \star x_{s_k}$ $\quad\backslash\backslash\, x_0 := \tilde{x}$
$\quad\quad t_k := h_k \cdot h_{s_k}$ $\quad\backslash\backslash\, h_0 := e$
$\quad (\mathsf{pk}, \mathsf{sk}) := ((x_k)_{k\in[m]}, (t_k)_{k\in[m]})$
$\quad (h, s) := ((h_k)_{k\in[m]}, (s_k)_{k\in[m]})$
$\quad \mathcal{H}[i] := (\mathsf{pk}, \mathsf{sk}, (h, s))$
$b \xleftarrow{\$} \{0, 1\}$
$b' \leftarrow \mathcal{A}^{\mathrm{O},\mathsf{H}}(\mathsf{pp})$ $\quad\backslash\backslash\, G_1$-$G_6$
$b' \leftarrow \mathcal{A}^{\mathrm{O},\mathsf{H}'}(\mathsf{pp})$ $\quad\backslash\backslash\, G_7$
return $[\![b = b']\!]$

Oracle $\textsc{Test}(\mathsf{ID}_1, \mathsf{ID}_2)$

if $L[\mathsf{ID}_1] = \bot \vee L[\mathsf{ID}_2] = \bot \vee \mathsf{ID}_1 = \mathsf{ID}_2$
$\quad$ **return** $\bot$
if $K[\mathsf{ID}_1, \mathsf{ID}_2] \neq \bot$
$\quad$ **return** $K[\mathsf{ID}_1, \mathsf{ID}_2]$
$(l_1, \mathsf{pk}_1, \mathsf{sk}_1) := L[\mathsf{ID}_1]$, $(l_2, \mathsf{pk}_2, \mathsf{sk}_2) := L[\mathsf{ID}_2]$
if $l_1 \neq \mathit{honest} \vee l_2 \neq \mathit{honest}$
$\quad$ **return** $\bot$
$K_0 := \mathsf{NIKE.SharedKey}(\mathsf{ID}_1, \mathsf{pk}_1, \mathsf{ID}_2, \mathsf{sk}_2)$ $\quad\backslash\backslash\, G_1$-$G_3$
$K_0 := \mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ $\quad\backslash\backslash\, G_4$-$G_7$
$K_1 \xleftarrow{\$} \{0, 1\}^\kappa$
$K[\mathsf{ID}_1, \mathsf{ID}_2] := K_b$
return $K[\mathsf{ID}_1, \mathsf{ID}_2]$

Oracle $\textsc{RegisterHonest}(\mathsf{ID})$

if $L[\mathsf{ID}] = (\mathit{corrupt}, *, \bot)$ **return** $\bot$
$(\mathsf{pk}, \mathsf{sk}, *) := \mathcal{H}[\mathsf{ctr}]$ $\quad\backslash\backslash\, G_2$-$G_7$
$\mathsf{ctr} := \mathsf{ctr} + 1$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{NIKE.Gen}(\mathsf{pp}, \mathsf{ID})$ $\quad\backslash\backslash\, G_1$
$L[\mathsf{ID}] := (\mathit{honest}, \mathsf{pk}, \mathsf{sk})$
return $(\mathsf{ID}, \mathsf{pk})$

Oracle $\textsc{CorruptReveal}(\mathsf{ID}_1, \mathsf{ID}_2)$

if $(L[\mathsf{ID}_1] = \bot \vee L[\mathsf{ID}_2] = \bot)$ **return** $\bot$
$(l_1, \mathsf{pk}_1, \mathsf{sk}_1) := L[\mathsf{ID}_1]$, $(l_2, \mathsf{pk}_2, \mathsf{sk}_2) := L[\mathsf{ID}_2]$
if $l_1 = l_2$ **return** $\bot$
Let $\mathsf{pk}_i \in \mathcal{H}$ $\quad\backslash\backslash\, G_4$-$G_7$
if $(\mathsf{pk}_{3-i}, *) \notin \mathcal{H}$
$\quad$ **return** $\mathsf{H}_1(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$
if $(\mathsf{pk}_{3-i}, *) \in \mathcal{H}$
$\quad$ **return** $\mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$
Let $\mathsf{sk}_i \neq \bot$
for $(j, k) \in [m]^2$
$\quad z_{j,k} := \mathsf{sk}_i[k] \star \mathsf{pk}_{3-i}[j]$
return $\mathsf{H}(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, (z_{j,k})_{j,k\in[m]})$

Oracle $\mathsf{H}(M)$ $\quad\backslash\backslash\, G_3$-$G_7$

Let $M = (\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z_{1,1}, \ldots, z_{m,m})$
if $(\mathsf{pk}_i, *, *) \in \mathcal{H} \wedge (\mathsf{pk}_{3-i}, *, *) \notin \mathcal{H}$ $\backslash\backslash\, i \in \{1, 2\}$, wlog $i = 1$
$\quad$ Let sk_1 s.t. $(\mathsf{pk}_1, \mathsf{sk}_1, *) \in \mathcal{H}$ $\quad\backslash\backslash\, G_3$-$G_5$
$\quad$ **for** $(j, k) \in [m]^2$
$\quad\quad z'_{j,k} := \mathsf{sk}_1[k] \star \mathsf{pk}_2[j]$
$\quad$ **if** $(z'_{1,1}, \ldots, z'_{m,m}) = (z_{1,1}, \ldots, z_{m,m})$
$\quad\quad$ **return** $\mathsf{H}_1(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$
$\quad$ **if** $\mathsf{TDT}(\mathsf{pk}_1, \mathsf{pk}_2, z_{1,1}, \ldots, z_{m,m}) = 1$ $\quad\backslash\backslash\, G_6$-$G_7$
$\quad\quad$ **return** $\mathsf{H}_1(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$
if $(\mathsf{pk}_1, *, *) \in \mathcal{H} \wedge (\mathsf{pk}_2, *, *) \in \mathcal{H}$
$\quad$ Let sk_1 s.t. $(\mathsf{pk}_1, \mathsf{sk}_1, *) \in \mathcal{H}$ $\quad\backslash\backslash\, G_3$-$G_5$
$\quad$ **for** $(j, k) \in [m]^2$
$\quad\quad z'_{j,k} := \mathsf{sk}_1[k] \star \mathsf{pk}_2[j]$
$\quad$ **if** $(z'_{1,1}, \ldots, z'_{m,m}) = (z_{1,1}, \ldots, z_{m,m})$
$\quad\quad$ **return** $\mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$
$\quad$ **if** $\mathsf{TDT}(\mathsf{pk}_1, \mathsf{pk}_2, z_{1,1}, \ldots, z_{m,m}) = 1$ $\quad\backslash\backslash\, G_6$-$G_7$
$\quad\quad$ **return** $\mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ $\quad\backslash\backslash\, G_6$
$\quad\quad$ **return** $\mathsf{H}'_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$ $\quad\backslash\backslash\, G_7$
return $\mathsf{H}_2(M)$

Trapdoor Test $\mathsf{TDT}(\mathsf{pk}_1, \mathsf{pk}_2, z_{1,1}, \ldots, z_{m,m})$

Let $\mathsf{sk}_1, (h, s)$ s.t. $(\mathsf{pk}_1, \mathsf{sk}_1, (h, s)) \in \mathcal{H}$
for $j \in [m]$
$\quad$ **for** $k \in [\ell, m]$
$\quad\quad$ **if** $z_{j,k} \neq h[k] \star z_{s[k]}$ $\quad\backslash\backslash\, z_0 := \mathsf{pk}_2[j]$
$\quad\quad\quad$ **return** 0
return 1

Fig. 23. Games G_1-G_7 for the proof of Theorem 7. O denotes the set of all CKS oracles, i.e. $\mathrm{O} = \{\textsc{RegisterHonest}, \textsc{RegisterCorrupt}, \textsc{CorruptReveal}, \textsc{Test}\}$, where $\textsc{RegisterCorrupt}$ is defined as in the original game in Figure 4. $\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3$ and H'_3 are internal random oracles. For improved readability we assume wlog $\mathsf{ID}_1 < \mathsf{ID}_2$. For G_1 and G_2, H is identical to H_2.

store all h_k and s_k in $\mathcal{H}$ so that we can use them in the trapdoor test. This is again a purely syntactical change as shown in Lemma 3. Therefore $\Pr[G_3^{\mathcal{A}} \Rightarrow 1] = \Pr[G_4^{\mathcal{A}} \Rightarrow 1]$.

$\textsc{Game}$ G_6. Next, we switch the check in H whether to use H_1 or H_2 to the trapdoor test from Lemma 3. Note that although we use the trapdoor test m times per hash query, the adversary only gets any information if *all* trapdoor tests succeed, so the number of trapdoor tests performed per hash query is irrelevant for the adversaries advantage. Therefore, by eq. (1) of Lemma 2,we can bound the difference between the two games as

$$\begin{aligned}|\Pr[\mathrm{G}_5^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_6^{\mathcal{A}} \Rightarrow 1]| &\leq \mathsf{Adv}^{\mathsf{TDT}}_{\mathsf{EGA},q,d,\ell,m}(\mathcal{D}) \\ &\leq 4\sqrt{\frac{(d+1)q}{\ell!\ell^{m-2\ell+1}}}\,,\end{aligned}$$

for any quantum distinguisher $\mathcal{D}$. The last inequality uses the bound from Lemma 3. Note that in G_6, the secret key is not needed anywhere.

Game G_7. In this game we reprogram the random oracle H on *all potential test queries* by reprogramming H on the set $\mathcal{S} := \{(*, *, \mathsf{pk}_1, \mathsf{pk}_2, (\mathsf{sk}_1[k] \star \mathsf{pk}_2[j])_{j,k\in[m]^2}) \mid \text{for } i, j \in [q_O]\colon (\mathsf{pk}_1, \mathsf{sk}_1) = \mathcal{H}[i], (\mathsf{pk}_2, *) = \mathcal{H}[j] \text{ and } i \neq j\}$ with uniformly random values from $\{0,1\}^\kappa$. Note that this reprogramming does not affect inputs which get forwarded to H_1 by H, since the reprogramming happens on values where both keys are in the honest set, which are forwarded to H_3. Therefore, we derive the reprogrammed values using another internal random oracle called H_3'. Note that Test and CorruptReveal still use H_3. So intuitively, the only way to distinguish the two games is to make a query to H which results in a query to the internal random oracle H_3', which contradicts the outputs of Test or CorruptReveal. However such a query includes valid $z_{i,j}$, which we enforce with the trapdoor test, and we can extract a GA-CDH solution from them.

Let H' be the reprogrammed random oracle, as described above. Since K_0 is now identically distributed to K_1 from the view of $\mathcal{A}$, we have that b is independent of $\mathcal{A}$'s view, thus

$$\Pr[\mathrm{G}_7^{\mathcal{A}} \Rightarrow 1] = \tfrac{1}{2}\,.$$

It remains to bound G_6-G_7. By the semi-classical O2H lemma, the difference between the games is bounded by the *classical event* Find when running $\mathcal{A}$ with the punctured oracle $\mathsf{H} \setminus \mathcal{S}$. The event Find is set to 1 if the semi-classical oracle measures that an input x lies in $\mathcal{S}$. The main difficulty when applying the lemma is to show how to simulate the membership testing $x \in \mathcal{S}$. We solve this issue using the trapdoor test in conjuction with the precomputed set $\mathcal{H}$. Let $\mathrm{G}_8^{\mathcal{A}}$ be as $\mathrm{G}_6^{\mathcal{A}}$ except that $\mathcal{A}$ gets access to $\mathsf{H} \setminus \mathcal{S}$, we have

$$\left|\Pr[\mathrm{G}_6^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_7^{\mathcal{A}} \Rightarrow]\right| \leq 2\sqrt{(d+1)\Pr[\textsc{Find}\colon \mathrm{G}_8^{\mathcal{A}}]}\,.$$

We bound the right-hand probability by the adversary $\mathcal{B}$ given in Figure 24. $\mathcal{B}$ obtains the challenge (y_0, y_1) and embeds these elements into the honest public keys as in G_6. It cannot compute the secret keys, but due to the changes in the previous games, it also does not need it. It additionally stores the bit c for each key, instead of the secret key. It simulates internal random oracles H_1, H_2, H_3 and H_3' using standard techniques and runs $\mathcal{A}^{\mathrm{O},\mathsf{H}\setminus\mathcal{S}}$ until the event Find $= 1$, then measures the corresponding input registers of $\mathsf{H} \setminus \mathcal{S}$ to obtain a set $\mathcal{T}$. It simulates all the oracles as in G_6, which $\mathcal{B}$ can do, since it can use the trapdoor test. Finally, once $\mathcal{T}$ is obtained, it searches for $(*, *, \mathsf{pk}_1, \mathsf{pk}_2, z_{1,1}, \ldots, z_{m,m}) \in \mathcal{T}$, where the first element of pk_1 was derived using random element $h[1]$ and base y_c and the first element of pk_2 was derived using random element $h'[1]$ and base y_{1-c}, and checks with the trapdoor test if the $z_{j,k}$ are correct. If this check passes, it returns $(h[1]^{-1} \cdot h'[1]^{-1}) \star z_{1,1}$ as the solution. First note that the bits c are information-theoretically hidden from the adversary. Thus, a test query will involve two public keys with different bits c with probability 1/2. Assuming that this is the case, $\mathcal{B}$ wins when Find $= 1$. This yields

$$\Pr[\textsc{Find}\colon \mathrm{G}_8^{\mathcal{A}}] \leq 2\mathsf{Adv}^{\mathsf{GA\text{-}CDH}}_{\mathsf{EGA}}(\mathcal{B})\,.$$

Adding up the analyzed bounds yields the claimed bound. The claimed running time of $\mathcal{B}$ follows from the running time of $\mathcal{A}$ and the analysis of the additional overhead is analogous to the proof of Theorem 5, which concludes our proof. □

E A Note on PSEC-KEM

Theorem 1 of [39] claims a QROM proof for CSIDH-PSEC-KEM from the GA-DDH assumption. However, there are multiple issues with the proof which boil down to the following three points:

Adversary $\mathcal{B}(y_0, y_1)$

$L[\mathcal{ID}] := \bot$
$K[\mathcal{ID}, \mathcal{ID}] := \bot$
$\mathsf{ctr} := 0$
$\mathcal{H} := \bot$
for $i \in [q_O]$
  $c \xleftarrow{\$} \{0,1\}$
  for $k \in [m]$
    $h_k \xleftarrow{\$} \mathcal{G}$
  $x_1 := h_1 \star y_c$
  for $k \in [2, \ell-1]$
    $x_k := h_k \star \tilde{x}$
  for $k \in [\ell, m]$
    $\backslash\backslash$ Enforce $\{s_\ell, \ldots, s_{2\ell-1}\} = [0, \ell-1]$
    $s_k \xleftarrow{\$} [0, \ell-1]$
    $x_k := h_k \star x_{s_k}$ $\backslash\backslash\, x_0 := \tilde{x}$
  $\mathsf{pk} := (x_k)_{k\in[m]}$
  $(h, s) := ((h_k)_{k\in[m]}, (s_k)_{k\in[m]})$
  $\mathcal{H}[i] := (\mathsf{pk}, c, (h, s))$
$b \xleftarrow{\$} \{0,1\}$
$\mathcal{T} \leftarrow$ Run $\mathcal{A}^{\mathrm{O},\mathsf{H}\setminus\mathcal{S}}(\mathsf{pp})$ until FIND and measure query register inputs
return $\mathsf{FindSolution}(\mathcal{T}, \mathcal{H})$

Oracle $\mathrm{O}^{\mathsf{SC}}_{\mathcal{S}}(|\psi_{in}, 0\rangle)$

$b := 0$
Parse ψ_{in} as $(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z_{1,1}, \ldots, z_{m,m})$
$b \leftarrow$ Measure [
  $\exists i, j \in [q_O] \wedge i \neq j, \exists(*, *, (h, s)) \in \mathcal{H}$:
  $[\![(\mathsf{pk}_1, *, (h, s)) = \mathcal{H}[i] \wedge (\mathsf{pk}_2, *, *) = \mathcal{H}[j]]\!]$
  $\wedge\, [\![\mathsf{TDT}(\mathsf{pk}_1, \mathsf{pk}_2, z_{1,1}, \ldots, z_{m,m}) = 1]\!]]$
return $(|\psi'_{in}\rangle, b)$

Oracle $\mathsf{H} \setminus \mathcal{S}(|\psi_{in}, \psi_{out}\rangle)$

$|\psi'_{in}, b\rangle \leftarrow \mathrm{O}^{\mathsf{SC}}_{\mathcal{S}}(|\psi_{in}, 0\rangle)$
if $b = 1$
  FIND $:= 1$
return $U_{\mathsf{H}}(|\psi'_{in}, \psi_{out}\rangle)$

Oracle $\mathsf{H}(M)$

Let $M := (\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z_{1,1}, \ldots, z_{m,m})$
if $(\mathsf{pk}_i, *, *) \in \mathcal{H} \wedge (\mathsf{pk}_{3-i}, *, *) \notin \mathcal{H}$ $\backslash\backslash\, i \in \{1,2\}$, wlog $i = 1$
  if $\mathsf{TDT}(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z_{1,1}, \ldots, z_{m,m}) = 1$
    return $\mathsf{H}_1(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$
if $(\mathsf{pk}_1, *, *) \in \mathcal{H} \wedge (\mathsf{pk}_2, *, *) \in \mathcal{H}$
  if $\mathsf{TDT}(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z_{1,1}, \ldots, z_{m,m}) = 1$
    return $\mathsf{H}_3(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2)$
return $\mathsf{H}_2(M)$

Function $\mathsf{FindSolution}(\mathcal{T}, \mathcal{H})$

$\mathcal{H}_0 := \{(a, 0, b) \in \mathcal{H}\}$
$\mathcal{H}_1 := \{(a, 1, b) \in \mathcal{H}\}$
for $(\mathsf{ID}_1, \mathsf{ID}_2, \mathsf{pk}_1, \mathsf{pk}_2, z_{1,1}, \ldots, z_{m,m}) \in \mathcal{T}$
  if $\exists (h, s), (h', s')$:
  $(\mathsf{pk}_1, c, (h, s)) \in \mathcal{H}_c \wedge (\mathsf{pk}_2, 1-c, (h', s')) \in \mathcal{H}_{1-c}$
    if $\mathsf{TDT}(\mathsf{pk}_1, \mathsf{pk}_2, z_{1,1}, \ldots, z_{m,m}) = 1$
      return $(h[1]^{-1} \cdot h'[1]^{-1}) \star z_{1,1}$
return $\bot$

Fig. 24. Adversary $\mathcal{B}$ against GA-CDH. O denotes the set of all CKS oracles, i.e. O = {REGISTERHONEST, REGISTERCORRUPT, CORRUPTREVEAL, TEST}, which are defined as in G_6 of Figure 23. TDT is also defined as in Figure 23.

1. The collision bounds used in their game hops are not compatible with known, optimal lower bounds for quantum computing. Specifically, second preimage collisions always loose a quadratic term, while full collision resistance loose a cubic term. While it is unclear which of the two is applicable to their proof, they claim a linear loss due to one of the two possibilities in their games, which contradicts known results in the quantum setting.
2. It analyzes classical events on the random oracle queries for example in hybrids 1 and 2. Such an approach will not work since a single uniform superposition query would trigger all events (informally speaking). Without these steps, it is unclear how to simulate decapsulation queries. For example, the equivalent of not querying an information-theoretically hidden value in the QROM with high probability amplitude would be done by using either the O2H lemma, or reducing to the Generic Search/Distinguishing Problem. Neither of these is considered in their proof.
3. It uses the semi-constant distribution technique from [41] which is usually used to prove security of signature schemes and identity-based encryption. The tool is used in the context of Hash-and-Sign signatures to be able to simulate signature queries and extract the solution to some underlying problem from the signature forgery. This is similarly the case for IBEs, since they inherently have an underlying signature scheme which can be derived from the IBE. To prove PKE/KEMs/AKEs secure, one usually uses the O2H lemma to reprogram the random oracle, but the paper claims that it is not possible to extract a GA-CDH solution from the random oracle queries due to the no-cloning principle and therefore uses the GA-DDH assumption. This is contradicted by the fact that one can prove GA-HEG to be IND-CPA secure using the O2H lemma similarly to our proofs, which also extracts a CDH value from a random oracle.

Remark 9. Additionaly, we remark that CSIDH-PSEC-KEM is essentially (up to one simplification) derived by applying the FO with explicit rejection to Hashed ElGamal. The FO Transform *with Explicit Rejection* has only been proven in recent work in the QROM in [16,41], using advanced techniques with careful analysis. We also use those for the proof of GA-HEG-KC. We believe that using those also for CSIDH-PSEC-KEM should yield a security proof from the GA-CDH assumption.