On the Impossibility of Constructing Efficient Key Encapsulation and Programmable Hash Functions in Prime Order Groups[*]

Goichiro Hanaoka, Takahiro Matsuda, and Jacob C.N. Schuldt

Research Institute for Secure Systems,
National Institute of Advanced Industrial Science and Technology
{hanaoka-goichiro, t-matsuda, jacob.schuldt}@aist.go.jp

Abstract. In this paper, we focus on the problem of minimizing ciphertext overhead, and discuss the (im)possibility of constructing key encapsulation mechanisms (KEMs) with low ciphertext overhead. More specifically, we rule out the existence of algebraic black-box reductions from the (bounded) CCA security of a natural class of KEMs to any non-interactive problem. The class of KEMs captures the structure of the currently most efficient KEMs defined in standard prime order groups, but restricts an encapsulation to consist of a single group element and a string. This result suggests that we cannot rely on existing techniques to construct a CCA secure KEM in standard prime order groups with a ciphertext overhead lower than two group elements. Furthermore, we show how the properties of an (algebraic) programmable hash function can be exploited to construct a simple, efficient and CCA secure KEM based on the hardness of the decisional Diffie-Hellman problem with the ciphertext overhead of just a single group element. Since this KEM construction is covered by the above mentioned impossibility result, this enables us to derive a lower bound on the hash key size of an algebraic programmable hash function, and rule out the existence of algebraic (poly, n)-programmable hash functions in prime order groups for any integer n. The latter result answers an open question posed by Hofheinz and Kiltz (CRYPTO'08) in the case of algebraic programmable hash functions in prime order groups.

Keywords: key encapsulation, chosen ciphertext security, programmable hash functions, algebraic black-box reductions

1 Introduction

The development of efficient and secure public key encryption has long been a central research area in cryptography, and in particular, achieving security against chosen ciphertext attacks (CCA security) while maintaining practical efficiency has been the focus of many papers in the literature. One of the most commonly used performance measures for public key encryption schemes, and the measure that we are going to focus on in this paper, is ciphertext overhead which expresses the additional cost in terms of storage and bandwidth when operating with encrypted data as opposed to unencrypted data.

The currently most efficient encryption schemes are based on hybrid encryption [14]. In this approach, a public key component, referred to as a key encapsulation mechanism (KEM), is used to encapsulate (i.e. encrypt) a random session key, and the actual message is then encrypted using a symmetric cipher which is referred to as a data encapsulation mechanism (DEM). The ciphertext overhead of this type of construction is dominated by the KEM. More specifically, if the KEM achieves CCA security, a redundancy-free DEM can be used since, in this case, only one-time CCA security is required of the DEM to obtain a CCA secure hybrid encryption scheme (i.e. a strong pseudo-random permutation can be used as a DEM [38]). If the KEM achieves the slightly weaker notion of constrained CCA security [28], an authenticated DEM is required, which will introduce an additional overhead of a message authentication code (MAC) assuming the authenticated DEM is

[*] An extended abstract of this paper will appear in the proceedings of CRYPTO 2012.

Table 1. The currently most efficient KEMs in terms of ciphertext overhead. The scheme Kiltz [33]† is identical to Kiltz [33], except that no hash function is used to derive the session-key (see Sect. 4.2 of [33]). In the table, CCCA denotes constrained CCA [28] and q-CCA denotes q-bounded CCA [13]. Furthermore, DDH denotes the decisional Diffie-Hellman assumption, CDH the computational Diffie-Hellman assumption, GDH the gap Diffie-Hellman assumption, GHDH the gap hashed Diffie-Hellman assumption, and DBDH the decisional bilinear Diffie-Hellman assumption.

Scheme	Security	Hardness assumption	Ciphertext overhead
CS [14]	IND-CCA	DDH	$3\vert\mathbb{G}\vert$
HaKu [22, Sect. 5]	IND-CCA	CDH	$3\vert\mathbb{G}\vert$
HaKu [22, Sect. 4.1]	OW-CCA	CDH	$3\vert\mathbb{G}\vert$
KD [34]	IND-CCCA	DDH	$2\vert\mathbb{G}\vert$
HoKi [28]	IND-CCCA	DDH	$2\vert\mathbb{G}\vert$
HaKu [22, Sect. 6]	IND-CCCA	DDH	$2\vert\mathbb{G}\vert$
Kiltz [33]	IND-CCA	GHDH	$2\vert\mathbb{G}\vert$
Kiltz [33]†	OW-CCA	GDH	$2\vert\mathbb{G}\vert$
BMW [8]	IND-CCA	DBDH	$2\vert\mathbb{G}\vert$
CHH+ [13]	IND-q-CCA	DDH	$1\vert\mathbb{G}\vert$

implemented using the encrypt-then-mac approach [3]. Alternatively, the KEM can be generically converted to a CCA secure KEM [2, 24], but this will likewise introduce an additional overhead of a MAC. Hence, when comparing the ciphertext overhead of CCA secure and constrained CCA secure KEMs, this additional overhead should be taken into account.

State-of-the-art Schemes. The currently most efficient (constrained) CCA secure KEMs, which are provably secure in the standard model, are defined in prime order groups [14, 34, 8, 33, 28, 10, 22, 25, 42, 24]. The ciphertext overhead in these schemes consists of at least two group elements. Note that for KEMs defined in prime order groups, each group element in the ciphertext overhead will contribute with at least 2λ bits for a security level of λ bits, since the order p of the group will have to satisfy $p > 2^{2\lambda}$ to prevent generic attacks against the underlying assumptions of the security of the KEMs. On the other hand, an additional overhead of a MAC, which must be taken into account when considering constrained CCA secure schemes, contributes with only λ bits.

Table 1 shows an overview of the security level achieved, security assumption used, and the ciphertext overhead of the currently most efficient schemes in terms of ciphertext overhead. Note that to obtain a ciphertext overhead of just two group elements while maintaining full CCA security, the current schemes require either interactive assumptions, e.g. Kiltz [33], or special groups equipped with a pairing, e.g. Boyen-Mei-Waters [8]. Furthermore, note that the DDH-based KEM by Cramer et al. [13] achieves a ciphertext overhead of just a single group element, but can only be shown to be q-bounded CCA secure (for a predetermined number of decryption queries q).

Besides being defined in prime order groups, the schemes in Table 1 share some structural properties. More specifically, all of the schemes include a random group element as part of the ciphertext which will be used to derive the key in the decapsulation. The remaining element(s) (except in the scheme KD [34], but see the footnote[1]) are used to decide whether the ciphertext should be accepted as "valid" or not, but does not otherwise contribute to the computation of the decapsulated key.

More Efficient Schemes? Given the existing KEMs, it would be natural to ask: *Is it possible to construct a CCA secure KEM with a ciphertext overhead of less than two group elements?*

[1] While the KEM component of the original Kurosawa-Desmedt scheme in [34] makes use of "implicit rejection" and does not explicitly check the validity of a KEM ciphertext, it is relatively straightforward to make the scheme use explicit rejection through a validity check. This KEM will be IND-CCCA secure under the DDH assumption, and will fit the description mentioned here (i.e. the session-key will only depend on the random group element in the corresponding ciphertext).

Considering the structure of the currently most efficient schemes mentioned above, one might consider implementing a more space-efficient validity check, using MACs and hash functions, as a potential strategy for reducing the ciphertext overhead in these schemes.

To illustrate this approach, consider the KEM by Cramer and Shoup [14]. In this scheme, a public key is of the form $pk = (g, X_1, X_2 = g^{x_1}X_1^{x_2}, X_3 = g^{y_1}X_1^{y_2}, X_4 = g^z)$, where g, X_1 are group generators, and the private key is $sk = (x_1, x_2, y_1, y_2, z)$. A ciphertext is of the form $(c_1 = g^r, c_2 = X_1^r, c_3 = X_2^r X_3^{r\alpha})$ and the corresponding key is $K = X_4^r$, where r is picked at random from $\mathbb{Z}_p$, p is the order of the group, $\alpha = H(c_1, c_2)$, and H is a target collision resistant hash function. The decapsulation first checks if

$$c_1^{x_1+y_1\alpha}c_2^{x_2+y_2\alpha} = c_3,$$

and if this is the case, outputs the session-key $K = c_1^z$. Otherwise, the ciphertext is rejected.

To reduce the ciphertext overhead, we might consider a slightly modified scheme in which the validity check is performed on the hash of the group elements instead of on the group elements themselves i.e. a ciphertext is of the form $(c_1 = g^r, c_2 = X_1^r, c_3' = H(X_2^r X_3^{r\alpha}))$, and the validity check is implemented as $H(c_1^{x_1+y_1\alpha}c_2^{x_2+y_2\alpha}) = c_3'$. Somewhat surprisingly, this leads to a CCA secure scheme as noted in [15]. This reduces the ciphertext overhead to match that of the other schemes defined in standard prime order groups and based on non-interactive assumptions, when taking into account the additional overhead of a MAC required by these schemes e.g. Kurosawa-Desmedt [34], Hofheinz-Kiltz [28] and Hanaoka-Kurosawa [22, Sect. 6].

A similar approach can be used to reduce the ciphertext overhead of the schemes Hofheinz-Kiltz [28], Hanaoka-Kurosawa [22], and Kurosawa-Desmedt [34] with explicit rejection. This yields KEMs with the ciphertext overhead of just a single group element and a hash value, which is lower than the currently most efficient schemes. However, unlike the modified version of the Cramer-Shoup scheme, the security proofs of these schemes do not immediately extend to the modified versions. Hence, it is not obvious what level of security these schemes provide.

1.1 Our Contribution

In this paper, we discuss the impossibility of constructing CCA secure KEMs in standard prime order groups with low ciphertext overhead. More specifically, as our main result, we show that it is impossible to construct an algebraic black-box reduction from the q-bounded one-way non-adaptive CCA security (`OW-`n`-CCA1`) of a class of KEMs in which the ciphertext consists only of a single (random) group element and a string, to the hardness of *any non-interactive problem* defined in the group, where n is the number of group elements in the public key of the KEM. Since the majority of standard model security reductions are algebraic black-box reductions, this result sheds light on the question regarding the minimal ciphertext overhead achievable while maintaining CCA security by ruling out a natural class of KEMs with similar structure to the currently most efficient KEMs defined in standard prime order groups. Furthermore, the result holds even for security against adversaries who are restricted to make a single parallel decryption query. Hence, we can additionally rule out the existence of algebraic black-box reductions from the non-malleability of the captured KEMs due to the implications shown in [4, 26]. These results imply that the approach of minimizing ciphertext overhead in the above mentioned schemes [28, 22, 34], by compressing group elements using a target collision resistant hash function (or a similar primitive), will not yield CCA secure or non-malleable KEMs based on non-interactive assumptions. Additionally, since the DDH-based KEM by Cramer et al. [13] is contained in the KEM class, the results imply that this scheme cannot be shown fully CCA secure or non-malleable based on any non-interactive assumption. See Section 3 for a definition of the class of KEMs we consider, Section 4 for our main impossibility result, and Section 6 for further discussion about the implications and limitations of our results.

Secondly, we show a simple construction of a CCA-secure KEM using programmable hash functions introduced by Hofheinz and Kiltz [30]. These hash functions capture, to some extent, the "programmability" achieved by a random oracle, and have been shown useful, for example, in the construction of short signatures in the standard model [30, 27]. We show that an algebraic $(q, 1)$-programmable hash function allows the construction of a q-bounded CCA ($\texttt{IND-}q\texttt{-CCA2}$) secure KEM based on the hardness of the DDH problem, with a ciphertext overhead of just a single group element (see Section 5.1 for the definition of a (α, β)-programmable hash function). Since this construction is covered by the above impossibility result, we can derive a lower bound on the level of programmability provided by a hash function with a given hash key size. Specifically, we show that a hash function with n group elements in the hash key cannot be $(\alpha, 1)$-programmable for any $\alpha > n$. Furthermore, we rule out the existence of algebraic (poly, β)-programmable hash functions in prime order groups for any integer β. This result answers an open question posed by Hofheinz and Kiltz [30] in the case of algebraic programmable hash functions in prime order groups. We note that all known constructions of programmable hash functions are algebraic [30, 27], and that the properties of these hash functions suggest that this may be inherent [30, Sect. 1.5].

1.2 Used Techniques and Related Work

The type of impossibility results we show is commonly known as a *black-box separation* [31]. More specifically, a black-box reduction from the security of a cryptographic scheme to the hardness of a problem is an algorithm which, given (black-box) access to any successful adversary against the scheme, successfully breaks any instance of the problem. A black-box separation result shows that such a black-box reduction cannot exist.

Two main lines of techniques have been used for showing black-box separations: oracle separations [31, 39] and meta-reductions [7, 12]. The former technique is typically used to show separations between primitives, e.g. [31, 40, 19, 21, 20, 6, 32], and is based on showing the existence of an oracle under which the primitive acting as a building block exists, but any instantiation of the "target" primitive is broken. The latter technique is somewhat more direct, and aims at showing that if there exists a reduction which, for example, reduces the security of a primitive to a computational assumption, then there exists a meta-reduction which uses the reduction as a black-box to break a (possibly different) computational assumption. Meta-reductions have successfully been used in [7, 12, 36, 16, 18, 37, 1]. For an overview of these definitions and techniques, see [39, 43].

While we are not considering a primitive-to-primitive reduction, we make use of a variant of the oracle separation technique. In particular, we show the existence of a *distribution* of oracles under which, on average over the choice of oracle, the non-interactive problem remains hard, while the CCA security of any of the considered KEMs can be broken. We show that such a distribution of oracles is sufficient to rule out a fully black-box reduction (in the taxonomy by Reingold et al. [39]) from the security of a KEM in the considered class of KEMs, to the hardness of any non-interactive problem. Since almost all security reductions for cryptographic primitives are of this type, ruling out the existence of fully black-box reduction gives strong evidence that the currently used techniques are not sufficient to prove the security of the KEMs in question. Our proof techniques, especially our formal treatment of the distribution of oracles, might be of independent interest.

The type of (fully) black-box reductions we are going to consider are *algebraic* reductions [7, 12, 36, 1]. Essentially, the algebraic property requires that the reduction only creates group elements by means of the group operation, and does not map arbitrary bit strings to group elements e.g. by applying a hash function to some string to obtain a group element. More specifically, for an algebraic algorithm, it is required that it is possible to compute the representation of a group element output by the algorithm in terms of the group elements which is given as input. For example, if an algebraic algorithm takes as input the group elements g_1, g_2 and outputs the element h, it should be possible to compute x_1, x_2 such that $g_1^{x_1} \cdot g_2^{x_2} = h$, given access to the randomness used by the algorithm. As argued in previous papers [7, 1], considering algebraic reductions is not overly restrictive, and almost all known security reductions for CCA secure KEMs in the standard model are algebraic. In particular, the security reduction for the KEMs shown in Table 1 are all algebraic.

2 Preliminaries

In this section, we review our basic notation and several basic definitions. The standard definition of a KEM and the corresponding security notions (such as OW-n-CCA1, IND-n-CCA2 etc.), as well as the definition of concrete computational problems, are given in Appendix A.

2.1 Basic Notation

In this paper, we use the following basic notations and terminology. $\mathbb{N}$ denotes the set of all natural numbers, and if $n \in \mathbb{N}$ then $[n] = \{1, \ldots, n\}$. "PPTA" denotes a *probabilistic polynomial time algorithm.* If S is a set, "$x \leftarrow S$" denotes picking x uniformly from S. If $\mathbb{D}$ is a probability distribution, then "$x \leftarrow \mathbb{D}$" denotes choosing x according to $\mathbb{D}$, and $[\mathbb{D}]$ denotes the "support" of $\mathbb{D}$, that is, $[\mathbb{D}] = \{x | \Pr_{x' \leftarrow \mathbb{D}}[x' = x] > 0\}$. If $\mathcal{A}$ is a probabilistic algorithm then $y \leftarrow \mathcal{A}(x; r)$ denotes that $\mathcal{A}$ computes y as output by taking x as input and using r as randomness, and $\mathcal{A}^{\mathcal{O}}$ denotes that $\mathcal{A}$ has access to the oracle $\mathcal{O}$. Unless otherwise stated, λ denotes the security parameter. A function $f(\lambda) : \mathbb{N} \rightarrow [0, 1]$ is said to be *negligible* (resp. *noticeable*) if for all positive polynomials $p(\lambda)$ and all sufficiently large $\lambda \in \mathbb{N}$, we have $f(\lambda) < 1/p(\lambda)$ (resp. $f(\lambda) \geq 1/p(\lambda)$). Let $\mathbf{X}$ be a vector, then we denote by $\mathbf{X}[i]$ the i-th component of $\mathbf{X}$ and by $|\mathbf{X}|$ the length of $\mathbf{X}$.

2.2 Group Description

In this paper, we consider non-interactive problems with respect to a family of prime-order groups $\{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ indexed by the security parameter λ. For convenience, we consider the following PTA, which we call a *group scheme*, with which we will define such problems. (When λ is easily inferred from the context, we will usually leave out the subscript λ and just write $\mathbb{G}$.)

Definition 1 *A* group scheme GS *is a deterministic PTA which takes a security parameter* 1^λ *as input, and outputs a* group instance Λ *consisting of a description of a group* $\mathbb{G}$*, a prime* p *that corresponds to the order of the group* $\mathbb{G}$*, and a generator* $g \in \mathbb{G}$*. This process is denoted by* $\Lambda = (g, p, \mathbb{G}) \leftarrow \mathsf{GS}(1^\lambda)$*.*

Note that since GS is assumed to be deterministic, there is a one-to-one correspondence between a security parameter λ and the group description Λ generated by $\mathsf{GS}(1^\lambda)$.

2.3 Algebraic Algorithms

Intuitively, an algorithm $\mathcal{R}$ is called *algebraic* if there exists a corresponding algorithm, called the *extractor*, such that for any group element output by $\mathcal{R}$, the extractor can compute the representation of the group element with respect to the group elements given to $\mathcal{R}$ as input.

Formally, we adopt a similar approach to [1] and define the notion of algebraic algorithms as follows:

Definition 2 *Let* $\mathcal{R}$ *be a PPTA that takes* $\Lambda = (g, p, \mathbb{G})$ *(output by* GS*), a string* $aux \in \{0, 1\}^*$*, and group elements* $\mathbf{X} \in \mathbb{G}^n$ *for some* $n \in \mathbb{N}$ *as input, and outputs a group element* $Y \in \mathbb{G}$ *and a string* $ext \in \{0, 1\}^*$*.* $\mathcal{R}$ *is called* algebraic *with respect to* GS *if there exists a PPTA* $\mathcal{E}$ *receiving the same input as* $\mathcal{R}$ *(including the same random coins* r*) such that for any* $\Lambda \leftarrow \mathsf{GS}(1^\lambda)$*, all polynomial size* $\mathbf{X}$*, and any string* $aux \in \{0, 1\}^*$*, the following probability is negligible in* λ*:*

$$\Pr[(Y, ext) \leftarrow \mathcal{R}(\Lambda, \mathbf{X}, aux; r);\ (\mathbf{y}, ext) \leftarrow \mathcal{E}(\Lambda, \mathbf{X}, aux, r) : Y \neq \mathbf{X}^{\mathbf{y}}]$$

where the probability is over the choice of r *and the randomness used by* $\mathcal{E}$*, and* $\mathbf{X}^{\mathbf{y}}$ *is defined by* $\prod_{i \in [|\mathbf{X}|]} \mathbf{X}[i]^{\mathbf{y}[i]}$*.*

Note that the definition of an algebraic algorithm does not exclude the possibility that the auxiliary information aux contains group elements of $\mathbb{G}$, but the representation of the output element Y computed by the extractor must be with respect to $\mathbf{X}$.

Algebraic Oracle Algorithms. The above definition of algebraic algorithms can naturally be extended to algebraic "oracle" algorithms, which play an important role in our result.

Definition 3 *Let $\mathcal{R}$ be an oracle PPTA that takes $\Lambda = (g, p, \mathbb{G})$ (output by GS), a string $aux \in \{0,1\}^*$, and group elements $\mathbf{X} \in \mathbb{G}^n$ for some $n \in \mathbb{N}$ as input, and outputs a group element $Y \in \mathbb{G}$ and a string $ext \in \{0,1\}^*$. Furthermore, let q be the number of queries made by $\mathcal{R}$ to the oracle. We say that $\mathcal{R}$ is an* algebraic oracle algorithm *if there exists an algebraic algorithm $\widehat{\mathcal{R}}$ (which we denote the* decomposition *of $\mathcal{R}$) such that executing $\mathcal{R}^{\mathcal{O}}(\Lambda, \mathbf{X}, aux)$ is equivalent to performing the following sequence of computations: First set $\mathbf{X}_0 \leftarrow \mathbf{X}$ and $aux_0 \leftarrow aux$. Run $(\mathbf{Y}_1, (ext||\mathsf{st}_1)) \leftarrow \widehat{\mathcal{R}}(\Lambda, \mathbf{X}_0, aux_0)$ and repeat*

$$(\mathbf{X}'_i, aux'_i) \leftarrow \mathcal{O}(\Lambda, \mathbf{Y}_i, ext_i);\ \mathbf{X}_{i+1} \leftarrow (\mathbf{X}_i||\mathbf{X}'_i);\ aux_{i+1} \leftarrow (\mathsf{st}_i||aux'_i);$$
$$(\mathbf{Y}_{i+1}, (ext_{i+1}||\mathsf{st}_{i+1})) \leftarrow \widehat{\mathcal{R}}(\Lambda, \mathbf{X}_{i+1}, aux_{i+1})$$

for $i = 1, \ldots, q$. The last vector $\mathbf{Y}_{q+1}$ output by $\widehat{\mathcal{R}}$ is assumed to contain the single element Y.

Note that since the decomposition $\widehat{\mathcal{R}}$ of an algebraic oracle algorithm $\mathcal{R}$ is defined as an algebraic algorithm, the representation of any group element output by $\mathcal{R}$ or included in the oracle queries made by $\mathcal{R}$, can be calculated by appropriately using the extractor $\widehat{\mathcal{E}}$ for $\widehat{\mathcal{R}}$. In this case, the representation is with respect to all group elements that are given as input to $\mathcal{R}$ or returned in response to $\mathcal{R}$'s oracle queries.

The above definition can easily be extended to algorithms outputting multiple group elements. Note that we will regard any algorithm whose output does not contain any group elements, as an algorithm which outputs the "identity element" $1_{\mathbb{G}}$. Hence, this type of algorithm will also be considered to be algebraic.

2.4 Non-interactive Problems with Respect to a Prime Order Group

A non-interactive problem (NIP) P with respect to a group scheme GS consists of a tuple of algebraic PPTAs, $(\mathsf{I}, \mathsf{V}, \mathsf{U})$, that are defined as follows:

Instance generator I: This algorithm takes a group description $\Lambda = (g, p, \mathbb{G})$ (output from $\mathsf{GS}(1^\lambda)$) as input, and outputs a pair of a problem instance and a witness (y, w). (Without loss of generality we assume that the problem instance y contains the information on Λ.)

Verification algorithm V: This algorithm takes a problem instance y, and a string x, and a witness w as input (where (y, w) are output by $\mathsf{I}(\Lambda)$), and outputs $\top$ or $\bot$. We say that x is a valid solution to the problem instance y if $V(y, x, w) = \top$, and otherwise we say that x is an invalid solution.

Threshold algorithm U: This algorithm takes a problem instance y as input, and outputs a string x.

For a NIP $\mathsf{P} = (\mathsf{I}, \mathsf{V}, \mathsf{U})$ with respect to GS and an algorithm $\mathcal{A}$, define the experiment $\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}}(\lambda)$ as:

$$\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}}(\lambda) : \left[(g, p, \mathbb{G}) \leftarrow \mathsf{GS}(1^\lambda);\ (y, w) \leftarrow \mathsf{I}(\Lambda);\ x \leftarrow \mathcal{A}(1^\lambda, y);\ \text{Return 1 iff } \mathsf{V}(y, x, w) = \top\right]$$

Furthermore, define the threshold $\delta(\lambda) = \Pr[\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\mathsf{U}}(\lambda) = 1]$. Then, for an algorithm $\mathcal{A}$, we define the advantage of $\mathcal{A}$ in solving P by:

$$\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}}(\lambda) = \Pr[\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}}(\lambda) = 1] - \delta(\lambda)$$

Definition 4 *Let P be a NIP with respect to a group scheme GS. We say that P is* hard *if, for any PPTA $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that $\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}}(\lambda) \le \mu(\lambda)$.*

Intuitively, the algorithm U represents a trivial solution strategy for the problem P, and any successful algorithm is required to be better than this U. Typically, U always returns an invalid answer for "search" problems i.e. $\delta(\lambda) = 0$, and returns a random bit for "decision" problems i.e. $\delta(\lambda) = 1/2$.

The above definition of a non-interactive problem essentially captures all the non-interactive problems defined for prime order groups, which are used to prove the security of existing cryptographic primitives. Specifically, the definition includes the standard computational and decisional Diffie-Hellman problems as well as their q-type variants (q-SDH [5], q-ABDHE [17], etc.), and will even capture the CPA security of a KEM.

2.5 Algebraic Black-Box Reductions

In this paper, we will consider the following type of reduction algorithms which reduces the security of a KEM to the hardness of a non-interactive problem.

Definition 5 *Let* GS *be a group scheme, and let* Γ *be a KEM and* P *be a NIP with respect to* GS. *Furthermore, let* $\mathtt{GOAL\text{-}ATK}$ *be a security notion for a KEM. We say that there is an* algebraic black-box reduction *from the* $\mathtt{GOAL\text{-}ATK}$ *security of* Γ *to* P *if there exists an algebraic oracle PPTA* $\mathcal{R}$ *with the following property: For any (possibly computationally unbounded) algorithm* $\mathcal{A}$, *if* $\mathsf{Adv}^{\mathtt{GOAL\text{-}ATK}}_{\Gamma,\mathcal{A}}(\lambda)$ *is non-negligible, then so is* $\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{R}^{\mathcal{A}}}(\lambda)$.

We note that this type of reduction is categorized as a *"fully" black-box* reduction in the taxonomy by Reingold et al. [39]. In particular, note that the (algebraic) reduction algorithm is required to work universally for all successful (possibly inefficient) algorithms.

3 A Class of Simple and Space Efficient KEMs

The class of KEMs we consider essentially captures the structure of the existing KEMs defined in standard prime order groups like Cramer-Shoup [14], Kurosawa-Desmedt [34] (with explicit rejection), Hofheinz-Kiltz [28], Hanaoka-Kurosawa [22], and Cramer et al. [13], but requires the ciphertexts to consist of a single random group element and a string i.e. a ciphertext is required to be of the form $(g^r, \widetilde{f}(pk, r))$ where $r \leftarrow \mathbb{Z}_p$, p is the order of the group, pk is the public key of the scheme, $\widetilde{f} : \mathcal{PK} \times \mathbb{Z}_p \to \{0,1\}^*$ is a scheme-dependent function, and $\mathcal{PK}$ is the public key space.. This captures the approach highlighted in the introduction of replacing the group elements used for validity checking in the above KEMs with the output of a hash function or similar primitive. However, we note that $\widetilde{f}(pk, r)$ is not limited to be used in a potential validity check, but might also be used when deriving the session-key. The session-keys encapsulated by the ciphertexts are assumed to be group elements, but can be derived from all the information available in the encapsulation in any "algebraic" way. Note that this captures the key derivation in the above KEMs, and does not rule out the use of target collision resistant hash function, or the use of a Waters hash function [41] (and similar constructions).

We consider a class of KEMs $\mathcal{K}_{\mathsf{GS},n}$ defined with respect to a group scheme GS and a parameter $n \in \mathbb{N}$.

Definition 6 *A KEM* $\Gamma = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ *with respect to a group scheme* GS *belongs to* $\mathcal{K}_{\mathsf{GS},n}$, *where* $n \in \mathbb{N}$, *if it has the following properties:*

1. *The public key space* $\mathcal{PK}$ *is* $\{\Lambda\} \times \{0,1\}^{\mu(\lambda)} \times \mathbb{G}^n$, *where* μ *is a scheme-dependent function and* $\Lambda = (g, p, \mathbb{G}) \leftarrow \mathsf{GS}(1^\lambda)$ *i.e. a public key* pk *returned by* $\mathsf{KG}(\Lambda)$ *is of the form* $pk = (\Lambda, aux, X_1, \dots, X_n)$.
2. *A ciphertext and the corresponding session-key are of the form*

$$C = (c, d) = (g^r, \widetilde{f}(pk, r)) \in \mathbb{G} \times \{0,1\}^* \qquad \text{and} \qquad K = g^{f_0(pk,r)} \prod_{i \in [n]} X_i^{f_i(pk,r)} \in \mathbb{G}$$

where $r \leftarrow \mathbb{Z}_p$, *and* $\widetilde{f} : \mathcal{PK} \times \mathbb{Z}_p \to \{0,1\}^*$ *and* $f_0, \dots, f_n : \mathcal{PK} \times \mathbb{Z}_p \to \mathbb{Z}_p$ *are efficiently computable scheme-dependent functions.*

3. *For any $pk = (\Lambda, aux, X_1, \ldots, X_n) \in \mathcal{PK}$, the session-key obtained by decapsulating a ciphertext $C = (c, d)$ generated by $\mathsf{Enc}(pk)$ is of the form*
$$K = g^{\psi_0(pk,C,y_1,\ldots,y_n)} c^{\psi_1(pk,C,y_1,\ldots,y_n)} \tag{1}$$
where $\psi_i(pk, C, y_1, \ldots, y_n) = \psi_{i,0}(pk, C) + \sum_{j\in[n]} \psi_{i,j}(pk, C) \cdot y_j$ for $i \in \{0, 1\}$, $y_k = \log_g X_k$ for $k \in [n]$, and $\{\psi_{i,j} : \mathcal{PK} \times \mathbb{G} \times \{0, 1\}^* \to \mathbb{Z}_p\}_{i\in\{0,1\},j\in[n]}$ are efficiently computable scheme-dependent functions.
4. *For any $pk = (\Lambda, aux, X_1, \ldots, X_n) \in \mathcal{PK}$, the second component $d \in \{0, 1\}^*$ of a ciphertext $C = (c, d)$ generated by $\mathsf{Enc}(pk)$ can be re-computed as follows: $d = \widetilde{\psi}(pk, c, y_1, \ldots, y_n)$, where $y_i = \log_g X_i$ for $i \in [n]$ and $\widetilde{\psi} : \mathcal{PK} \times \mathbb{G} \times \mathbb{Z}_p^n \to \{0, 1\}^*$ is a scheme-dependent function.*

We would like to note the following:

– The values $y_1, \ldots, y_n$ defined above might not correspond to the private key of the scheme, and the decapsulation might not be done as shown in (1), but it is required that any valid session-key K output by Dec satisfy equation (1).
– If a KEM in $\mathcal{K}_{\mathsf{GS},n}$ satisfies correctness, it must hold that
$$f_0(pk, r) + \sum_{i\in[n]} f_i(pk, r) \cdot y_i = \psi_0(pk, C, y_1, \ldots, y_n) + r \cdot \psi_1(pk, C, y_1, \ldots, y_n)$$
for any $r \in \mathbb{Z}_p$. Hence, the requirement that the functions ψ_0 and ψ_1 are linear functions in $y_1, \ldots, y_n$ is arguably a very mild restriction.
– That the component d of a ciphertext can be recomputed using $\widetilde{\psi}$, is natural if d is used as a part of a validity check. Note, however, that no requirements are made regarding how a KEM in $\mathcal{K}_{\mathsf{GS},n}$ implements validity checking, and d might, for example, also be used to compute the session-key K.
– That aux and $d = \widetilde{f}(pk, r)$ are strings imply that the representation of a group element output by any algebraic algorithm taking a public key or a ciphertext as input, cannot depend on group elements derived from aux or d.
– There are no restrictions on the scheme-dependent functions $\widetilde{f}$, $\{f_i\}_{i\in\{0,\ldots,n\}}$, $\widetilde{\psi}$, $\{\psi_{i,j}\}_{i\in\{0,1\},j\in[n]}$ (other than that they are efficiently computable), and these might use non-linear functions like cryptographic hash functions and MACs, which is not allowed in "structure-preserving" encryption [9].

4 Main Impossibility Result

In this section, we will show the following theorem which captures our main result.

Theorem 7 *For any group scheme GS, for any KEM $\Gamma \in \mathcal{K}_{\mathsf{GS},n}$ where $n \in \mathbb{N}$, and for any NIP P with respect to GS, if P is hard, then there is no algebraic black-box reduction from the $\mathtt{OW}$-n-$\mathtt{CCA1}$ security of Γ to P.*

We will show this theorem by using a variant of the oracle separation technique [31, 39]. Specifically, the theorem follows from the following lemma (the full proof of this lemma can be found in Appendix B).

Lemma 8 *Let GS be a group scheme, $\Gamma \in \mathcal{K}_{\mathsf{GS},n}$ be a KEM where $n \in \mathbb{N}$, and P be a NIP with respect to GS. Furthermore, let $\mathtt{GOAL} \in \{\mathtt{OW}, \mathtt{IND}\}$ and $\mathtt{ATK} \in \{\mathtt{CPA}, q\text{-}\mathtt{CCA1}, \mathtt{CCA1}, q\text{-}\mathtt{CCA2}, \mathtt{CCA2}\}$ (with $q \in \mathbb{N}$). Assume there exists a distribution $\mathbb{D}$ of oracles $\mathcal{O}$ satisfying the following two conditions:*

1. *There exists an algebraic oracle PPTA $\mathcal{A}$ such that $\mathbf{E}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathtt{GOAL}\text{-}\mathtt{ATK}}_{\Gamma,\mathcal{A}^{\mathcal{O}}}(\lambda)]$ is non-negligible.*
2. *For any algebraic oracle PPTA $\mathcal{A}$, there exist PPTAs $\mathcal{B}_1$ and $\mathcal{B}_2$, a polynomial $Q(\lambda)$, and a negligible function $\mu(\lambda)$ such that*
$$\mathop{\mathbf{E}}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathcal{O}}}(\lambda)] \le Q(\lambda) \cdot \mathsf{Adv}^{\mathtt{DL}}_{\mathsf{GS},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{B}_2}(\lambda) + \mu(\lambda).$$

Then, if P is hard, there is no algebraic black-box reduction from the $\mathtt{GOAL}\text{-}\mathtt{ATK}$ security of Γ to P.

Proof Sketch. The lemma is proved by contradiction. We assume simultaneously that the NIP $\mathsf{P} = (\mathsf{I}, \mathsf{V}, \mathsf{U})$ is hard and that there is an algebraic black-box reduction from the `GOAL-ATK` security of the KEM Γ to P. The latter guarantees that there exists an algebraic oracle PPTA $\mathcal{R}$ that satisfies Definition 5. We consider two separate cases: the discrete logarithm (DL) problem with respect to GS is *hard*, and the DL problem with respect to GS is *not* hard. For each case we will show a contradiction.

The second case is fairly easy to show and does not require the use of the oracle $\mathcal{O}$. Specifically, that the DL problem is not hard implies the existence of an adversary $\mathcal{A}'$ that successfully breaks the KEM Γ in the sense of the `GOAL-ATK` security considered in the lemma, by simply recovering the randomness r from the challenge ciphertext. Then, the definition of $\mathcal{R}$ implies that $\mathcal{R}^{\mathcal{A}'}$ can solve P with non-negligible advantage. Here, note that $\mathcal{R}^{\mathcal{A}'}$ can be implemented by a single PPTA $\mathcal{R}'$ which internally runs $\mathcal{A}'$ since both $\mathcal{R}$ and $\mathcal{A}'$ are PPTAs. However, the existence of such $\mathcal{R}'$ contradicts that P is hard.

The first case, in which the DL problem is hard, has some similarities with the above case, but is more interesting and more involved. We make use of the oracle $\mathcal{O}$ chosen according to $\mathbb{D}$. The first condition of the lemma guarantees the existence of an algebraic oracle PPTA $\mathcal{A}$ which, given access to $\mathcal{O}$, has non-negligible "expected" advantage in breaking the `GOAL-ATK` security of the KEM Γ, where the expectation is over the choice of $\mathcal{O}$ according to $\mathbb{D}$.

Now, in order to reach a contradiction, we would like to construct an algebraic oracle PPTA $\widehat{\mathcal{R}}$ which, given access to $\mathcal{O}$ (chosen according to $\mathbb{D}$), has non-negligible "expected" advantage in solving P, by using $\mathcal{A}$ and the reduction algorithm $\mathcal{R}$ as building blocks. One might think that just running $\mathcal{R}^{\mathcal{A}^{\mathcal{O}}}$ on input the problem instance y given to $\widehat{\mathcal{R}}$ is enough for that purpose, but this is not the case. Recall that $\mathcal{R}$ is only guaranteed to work when $\mathcal{A}^{\mathcal{O}}$ successfully breaks the `GOAL-ATK` security of Γ. Furthermore, only the "expected" advantage of $\mathcal{A}$ is guaranteed to be non-negligible. Hence, there is a possibility that a particular choice of $\mathcal{O}$ is "bad," and that $\mathcal{A}$'s advantage under this $\mathcal{O}$ is not non-negligible. If the oracle $\mathcal{O}$ is bad, then nothing is guaranteed about the success probability of $\mathcal{R}^{\mathcal{A}^{\mathcal{O}}}$ in solving P. In particular, the advantage of $\mathcal{R}$ might even be negative, and the overall expected advantage in solving P might not be non-negligible.

To deal with this issue, $\widehat{\mathcal{R}}$ first tests whether the given oracle $\mathcal{O}$ is "good" by running $\mathcal{R}^{\mathcal{A}^{(\cdot)}}$ many times with independently generated problem instances. If the success probability of $\mathcal{R}^{\mathcal{A}^{\mathcal{O}}}$ (measured by $\widehat{\mathcal{R}}$) sufficiently exceeds the threshold $\delta(\lambda)$, then $\widehat{\mathcal{R}}$ labels the oracle "good" and runs $\mathcal{R}^{\mathcal{A}^{\mathcal{O}}}$ with the given instance y. Otherwise, $\widehat{\mathcal{R}}$ runs the threshold algorithm U on input y to avoid a heavy negative contribution to the expected advantage. Constructed as above, $\widehat{\mathcal{R}}$'s expected advantage (over the choice of $\mathcal{O}$ according to $\mathbb{D}$) can be shown to be non-negligible. Then, the existence of such an algebraic oracle algorithm, together with the second condition given in the lemma and the assumption that the DL problem is hard, implies the existence of a PPTA $\mathcal{B}_2$ that solves P with non-negligible advantage, which again contradicts that P is hard. Hence, we can conclude that either P is not hard or there exists no algebraic black-box reduction from the `GOAL-ATK` security of Γ to the hardness of P. $\square$

To make use of the above Lemma 8, we will first define a distribution of oracles, and then proceed to show that the conditions 1 and 2 are satisfied for this distribution. This will complete the proof of Theorem 7.

4.1 The Oracle and the Distribution

The oracle we consider is associated to a KEM which belongs to the class $\mathcal{K}_{\mathsf{GS},n}$. More specifically, for any KEM $\Gamma \in \mathcal{K}_{\mathsf{GS},n}$ and corresponding public key space $\mathcal{PK}$, consider an oracle $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2\}$ defined by a function $F : \mathcal{PK} \to \mathbb{Z}_p^n \times \{0,1\}^\lambda$ where n indicates the number of group elements in a public key of Γ (i.e. $pk = (\Lambda, aux, X_1, \ldots, X_n)$ and $\Lambda = (g, p, \mathbb{G}) \leftarrow \mathsf{GS}(1^\lambda)$):

- $\mathcal{O}_1$ takes as input a public key pk, and returns $\perp$ if $pk \notin \mathcal{PK}$. Otherwise, $\mathcal{O}_1$ computes $(r_1, \ldots, r_n, \sigma) \leftarrow F(pk)$ and the ciphertexts $\{(C_i, K_i) \leftarrow \mathsf{Enc}(pk; r_i)\}_{i\in[n]}$. Lastly, $\mathcal{O}_1$ returns $(C_1, \ldots, C_n, \sigma)$.

– $\mathcal{O}_2$ takes as input a public key pk, session-keys $(K_1, \ldots, K_n) \in \mathbb{G}^n$, and a tag $\sigma \in \{0,1\}^\lambda$. If $pk \notin \mathcal{PK}$, then $\mathcal{O}_2$ returns $\perp$. Otherwise, $\mathcal{O}_2$ computes $(r_1, \ldots, r_n, \sigma') \leftarrow F(pk)$ and $(C_i, K'_i) \leftarrow \mathsf{Enc}(pk; r_i)$ for each $i = 1, \ldots, n$, and then checks if $K_i = K'_i$ for all $i \in [n]$ and $\sigma = \sigma'$. If the check fails, then $\mathcal{O}_2$ returns $\perp$. Otherwise, $\mathcal{O}_2$ computes $\{u_i = \psi_1(pk, C_i, y_1, \ldots, y_n)\}_{i\in[n]}$ and returns these values, where ψ_1 is the scheme-dependent function of Γ.

Note that $\mathcal{O}$ defined above is deterministic.

By picking the function $F : \mathcal{PK} \to \mathbb{Z}_p^n \times \{0,1\}^\lambda$ uniformly at random from all possible functions with the proper domain and range, we obtain a distribution of the above defined oracles. In the following, this distribution will be denoted $\mathbb{D}$.

4.2 Breaking a KEM using the Oracle

We will now show that an oracle $\mathcal{O}$ chosen according to the distribution $\mathbb{D}$ defined in Section 4.1 can be used to break the `OW-`n`-CCA1` security of a KEM in $\mathcal{K}_{\mathsf{GS},n}$. The full proof of the following lemma can be found in Appendix C.

Lemma 9 *Let* GS *be a group scheme,* $\Gamma \in \mathcal{K}_{\mathsf{GS},n}$ *be a KEM where* $n \in \mathbb{N}$, *and let* $\mathbb{D}$ *be the distribution of the oracles* $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$ *described above. Then there exists an algebraic oracle PPTA* $\mathcal{A}$ *such that* $\mathbf{E}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathcal{O}}}(\lambda)] \geq \frac{3}{4n^2}$.

Proof Sketch. In the following, we consider the `OW-`n`-CCA1` experiment in which we take into account the choice of oracle $\mathcal{O}$ according to $\mathbb{D}$. Note that the success probability of an adversary $\mathcal{A}$ in this experiment will be given by $\mathbf{E}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathcal{O}}}(\lambda)]$.

Recall that the session-key K of a ciphertext $C = (c, d)$ of the KEM Γ is of the form $K = g^{\psi_0(pk,C,y_1,\ldots,y_n)} c^{\psi_1(pk,C,y_1,\ldots,y_n)}$ where $\psi_1(pk, C, y_1, \ldots, y_n) = \psi_{1,0}(pk, C) + \sum_{i\in[n]} \psi_{1,i}(pk, C) \cdot y_i$. Therefore, from a `OW-`$n$`-CCA1` adversary's viewpoint, the difficulty of recovering a session-key K from a ciphertext C must lie in the calculation of the component $c^{\sum_{i\in[n]} \psi_{1,i}(pk,C)\cdot y_i}$. However, we construct an algebraic oracle PPTA $\mathcal{A}$ that makes use of the oracle $\mathcal{O} \in [\mathbb{D}]$ and the decapsulation oracle $\mathcal{O}_{dec}$ to calculate this component for the challenge ciphertext C^*, and thereby break the `OW-`n`-CCA1` security of Γ. $\mathcal{A}$ is constructed as follows:

Given a public key $pk = (\Lambda, aux, X_1, \ldots, X_n)$ for Γ, $\mathcal{A}$ simply submits this to $\mathcal{O}_1$ to obtain n randomly generated ciphertexts $(C_1, \ldots, C_n)$ under pk, and a tag σ. Then $\mathcal{A}$ submits the ciphertext $(C_1, \ldots, C_n)$ to $\mathcal{O}_{dec}$ to obtain the corresponding decapsulations $(K_1, \ldots, K_n)$. Lastly, $\mathcal{A}$ will submit $(K_1, \ldots, K_n, \sigma)$ to $\mathcal{O}_2$ to obtain the values $\{u_j = \psi_{1,0}(pk, C_j) + \sum_{i\in[n]} \psi_{1,i}(pk, C_j) \cdot y_i\}_{j\in[n]}$, where $y_i = \log_g X_i$ for $i \in [n]$.

For a ciphertext C, let $\boldsymbol{\psi}(pk, C) = (\psi_{1,1}(pk, C), \ldots, \psi_{1,n}(pk, C)) \in \mathbb{Z}_p^n$ be a row vector. Furthermore, let $\mathbf{y}^{\mathrm{T}} = (y_1, \ldots, y_n) \in \mathbb{Z}_p^n$ and let $u'_j = u_j - \psi_{1,0}(pk, C_j)$ for all $j \in [n]$. Then, using the values returned by $\mathcal{O}_2$, $\mathcal{A}$ can construct a system of equations $\{\boldsymbol{\psi}(pk, C_j) \cdot \mathbf{y} = u'_j\}_{j\in[n]}$. If the vectors $\{\boldsymbol{\psi}(pk, C_j)\}_{j\in[n]}$ are linearly independent, this system will have the unique solution $\mathbf{y}^{\mathrm{T}} = (y_1, \ldots, y_n)$, and $\mathcal{A}$ will be able to recover this by solving the equation system. Obtaining $\mathbf{y}$ allows $\mathcal{A}$ to trivially calculate K^*.

The tricky part is the case in which the vectors $\{\boldsymbol{\psi}(pk, C_j)\}_{j\in[n]}$ are linearly dependent. Recall that the ciphertexts $(C_1, \ldots, C_n)$ are generated randomly by the oracle $\mathcal{O}_1$, and that the challenge ciphertext C^* is likewise randomly generated by the `OW-`n`-CCA1` experiment. The key observation, which we will show in the full proof, is that if the vectors $\{\boldsymbol{\psi}(pk, C_j)\}_{j\in[n]}$ are linearly dependent, then there is a high probability that the vector $\boldsymbol{\psi}(pk, C^*)$ is linearly dependent on the $n-1$ vectors $\{\boldsymbol{\psi}(pk, C_j)\}_{j\in[n-1]}$. Hence, $\boldsymbol{\psi}(pk, C^*) \cdot \mathbf{y} = \sum_{i\in[n]} \psi_{1,i}(pk, C^*) \cdot y_i$ can be represented as a linear combination of the $n - 1$ values $\{u'_j = \boldsymbol{\psi}(pk, C_j) \cdot \mathbf{y}\}_{j\in[n-1]}$, where the latter are known to $\mathcal{A}$. Therefore, in this case, $\mathcal{A}$ can calculate $\psi_1(pk, C^*) = \psi_{1,0}(pk, C^*) + \sum_{i\in[n]} \psi_{1,i}(pk, C^*) \cdot y_i$, from which $\mathcal{A}$ can recover K^*.

In the full proof, we will show that regardless of the probability that the n vectors $\{\boldsymbol{\psi}(pk, C_j)\}_{j\in[n]}$ are linearly dependent, $\mathcal{A}$ will have an expected success probability with the claimed lower bound. $\square$

Breaking Non-malleability. Inspecting the proof of the above lemma reveals that the n decapsulation queries made by the defined adversary $\mathcal{A}$ are independent i.e. the decapsulation queries can be made as a single parallel decapsulation query containing n ciphertexts. Since indistinguishability against a parallel chosen ciphertext attack is equivalent to the notion of non-malleability [4, 26, 35], this implies that the constructed adversary can be used to successfully attack the non-malleability of the KEM Γ as well. Hence, our impossibility result can easily be extended to rule out the existence of an algebraic black-box reduction from the non-malleability of a KEM $\Gamma \in \mathcal{K}_{\mathsf{GS},n}$ to a NIP P with respect to GS.

4.3 Simulating the Oracle while Solving a NIP

In this subsection, we will show that the oracles defined in Section 4.1 are essentially useless for an algebraic algorithm trying to solve a NIP. The full proof of the following lemma can be found in Appendix D.

Lemma 10 *Let GS be a group scheme, $\Gamma \in \mathcal{K}_{\mathsf{GS},n}$ be a KEM where $n \in \mathbb{N}$, and P be a NIP with respect to GS. Furthermore, let $\mathbb{D}$ be the distribution of the oracles $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$ (corresponding to Γ) defined as above. Then, for any algebraic oracle PPTA $\mathcal{A}$, there exist PPTAs $\mathcal{B}_1$ and $\mathcal{B}_2$, a polynomial $Q(\lambda)$, and a negligible function $\mu(\lambda)$ such that*

$$\mathop{\mathbf{E}}_{\mathcal{O} \leftarrow \mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathcal{O}}}(\lambda)] \leq Q(\lambda) \cdot \mathsf{Adv}^{\mathsf{DL}}_{\mathsf{GS},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{B}_2}(\lambda) + \mu(\lambda).$$

Proof Sketch. In the following, we consider the NIP hardness experiment $\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}}(\lambda)$ in which we take into account the choice of the oracle $\mathcal{O}$ according to $\mathbb{D}$. Note that the advantage of an adversary $\mathcal{A}$ in this experiment is given by $\mathbf{E}_{\mathcal{O} \leftarrow \mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathcal{O}}}(\lambda)]$.

To prove the lemma, we show that for any algebraic oracle PPTA $\mathcal{A}$ with access to $\mathcal{O}$ and which attempts to solve the NIP P, it is possible to construct another PPTA $\mathcal{B}_2$ which has almost the same advantage as $\mathcal{A}$ in solving the same P without access to $\mathcal{O}$.

More specifically, $\mathcal{B}_2$ will make use of $\mathcal{A}$ as a building block and simulate the oracle $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$ chosen according to $\mathbb{D}$ for $\mathcal{A}$. $\mathcal{B}_2$ takes a problem instance y (of P) as input, and generates an empty list L which is used to simulate $\mathcal{O}$. Then $\mathcal{B}_2$ picks randomness $r_{\mathcal{A}}$ and runs $\mathcal{A}$ with input y and randomness $r_{\mathcal{A}}$.

The main difficulty of simulating the oracle $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$ is that $\mathcal{A}$ may use $\mathcal{O}_1$ and $\mathcal{O}_2$ multiple times in any order. Recall, however, that when chosen according to $\mathbb{D}$, the function F used in $\mathcal{O}_1$ and $\mathcal{O}_2$ is a random function, and the tag $\sigma \in \{0,1\}^{\lambda}$, which is contained in the output of F, works like an information-theoretically secure MAC. Therefore, when $\mathcal{A}$ asks an $\mathcal{O}_2$-query with a fresh pk (that has not appeared in any of $\mathcal{A}$'s previous queries), $\mathcal{B}_2$ can immediately return $\perp$, which will be an almost perfect simulation of $\mathcal{O}_2$ for this type of query. $\mathcal{B}_2$ simulates $\mathcal{O}_1$ by "lazy-sampling" of the random function F and generating ciphertexts $\{C_i = (c_i, d_i)\}_{i \in [n]}$ using Enc. All values returned to $\mathcal{A}$ in an $\mathcal{O}_1$-query, as well as the encapsulated session-keys, are stored by $\mathcal{B}_2$ in the list L. Furthermore, when $\mathcal{A}$ makes a valid $\mathcal{O}_2$-query $(pk, K_1, \ldots, K_n, \sigma)$ (for which $\mathcal{O}_2$ will not return $\perp$), $\mathcal{B}_2$ can run the extractor corresponding to (the decomposition of) $\mathcal{A}$ to obtain values $\{u_i\}_{i \in [n]}$ such that $K_i = c_i^{u_i} g^{z_i}$ for some value $z_i \in \mathbb{Z}_p$ unknown to $\mathcal{B}_2$. Lastly, $\mathcal{B}_2$ returns $(u_1, \ldots, u_n)$. Whether the values $\{K_i\}_{i \in [n]}$ are correct decapsulation results can be checked using the list L. Note that since the randomness $r_{\mathcal{A}}$ is chosen by $\mathcal{B}_2$, $\mathcal{B}_2$ is able to run the extractor for $\mathcal{A}$.

Here, however, we have to be careful because the above simulation could fail if either of the following events occurs: (1) the extractor fails, or (2) there is an index $i \in [n]$ such that the extracted value u_i is different from $\psi_1(pk, C_i, y_1, \ldots, y_n)$. Fortunately, the probability of (1) occurring is negligible by the definition of an algebraic oracle algorithm. Moreover, the probability of (2) occurring for an index $i \in [n]$ in one of $\mathcal{A}$'s $\mathcal{O}_2$-queries can be bounded by the advantage $\mathsf{Adv}^{\mathsf{DL}}_{\mathsf{GS},\mathcal{B}_1}(\lambda)$ of another PPTA $\mathcal{B}_1$ which solves the DL problem. Put differently, $\mathcal{B}_2$ succeeds in simulating the oracle $\mathcal{O}$ for $\mathcal{A}$ almost perfectly. Furthermore, since $\mathcal{B}_2$ succeeds in solving P whenever $\mathcal{A}$ does, the lemma follows. □

5 Impossibility Results for Programmable Hash Functions

In this section, we show lower bounds on the "programmability" of an algebraic programmable hash functions defined in a prime order group. We will do this indirectly, by first showing how to construct a CCA secure KEM based on DDH, with a ciphertext consisting of just a single group element, from an algebraic programmable hash function. Since this KEM construction is captured by the class considered in the previous sections, we can derive the lower bounds by combining this with the previous impossibility result.

5.1 Programmable Hash Functions

Definition 11 *Let $\alpha, \beta \in \mathbb{N}$. A* (α, β)-programmable hash function *H with respects to a group scheme* GS *and with input length $\ell(\lambda)$, consists of the following four algorithms* $(\mathsf{HGen}, \mathsf{Eval}, \mathsf{HTrapGen}, \mathsf{HTrapEval})$

- HGen *takes a group description $\Lambda = (g, p, \mathbb{G})$ (output from $\mathsf{GS}(1^\lambda)$) and returns a hash key κ.*
- Eval *takes κ and a string $s \in \{0,1\}^{\ell(\lambda)}$ as input, and returns a group element of $\mathbb{G}$.*
- $\mathsf{HTrapGen}$ *takes Λ and group elements h_1, h_2 as input, and returns a hash key κ and a trapdoor τ.*
- *For all group elements $h_1, h_2 \in \mathbb{G}$, the statistical difference between the keys $\kappa \leftarrow \mathsf{HGen}(\Lambda)$ and the first component κ of the output from $\mathsf{HTrapGen}(\Lambda, h_1, h_2)$ is negligible.*
- *On input a string $s \in \{0,1\}^{\ell(\lambda)}$ and trapdoor τ,* $\mathsf{HTrapEval}$ *returns $a_s, b_s \in \mathbb{Z}_p$ such that* $\mathsf{Eval}(\kappa, s) = h_1^{a_s} h_2^{b_s}$.
- *For all group elements $h_1, h_2 \in \mathbb{G}$ and $(\kappa, \tau) \leftarrow \mathsf{HTrapGen}(\Lambda)$, and for all strings $s_1, \ldots, s_\alpha \in \{0,1\}^{\ell(\lambda)}$ and $s'_1, \ldots, s'_\beta \in \{0,1\}^{\ell(\lambda)}$ such that $s_i \neq s'_j$ for all i, j, we have*

$$\Pr[a_{s_1} = \cdots = a_{s_\alpha} = 0 \wedge a_{s'_1}, \ldots, a_{s'_\beta} \neq 0]$$

is noticeable in λ, where $(a_{s_i}, b_{s_i}) \leftarrow \mathsf{HTrapEval}(\tau, s_i)$, $(a_{s'_j}, b_{s'_j}) \leftarrow \mathsf{HTrapEval}(\tau, s'_j)$, and the probability is taken over the randomness used by $\mathsf{HTrapGen}$.

If H is $(q(\lambda), \beta)$-programmable for every polynomial $q(\lambda)$, we say that H is (poly, β)-programmable. Furthermore, we say that a programmable hash function is *algebraic* if all algorithms of H are algebraic algorithms[2].

In the following, we will make explicit use of the extractors for HGen and Eval. More specifically, let $\kappa \leftarrow \mathsf{HGen}(\Lambda)$ be given by $\kappa = (aux, X_1, \ldots, X_n)$ where $X_i \in \mathbb{G}$ for all $i \in [n]$ and it is assumed that aux does not contain any elements of $\mathbb{G}$. Let $h \in \mathbb{G}$ be an element returned by Eval on input a string s. Then, if HGen and Eval are algebraic algorithms, there exist extractors $\mathcal{E}_{\mathsf{HGen}}$ and $\mathcal{E}_{\mathsf{Eval}}$ with the following properties:

- On input $\Lambda = (g, p, \mathbb{G})$ and randomness r_{HGen} used to run HGen, $\mathcal{E}_{\mathsf{HGen}}$ returns values $(y_1, \ldots, y_n)$ such that $X_i = g^{y_i}$ for all $i \in [n]$.
- On input $\kappa = (aux, X_1, \ldots, X_n)$ and a string $s \in \{0,1\}^{\ell(\lambda)}$, $\mathcal{E}_{\mathsf{Eval}}$ returns values $(a_1, \ldots, a_n)$ such that $h = \prod_{i \in [n]} X_i^{a_i} = \mathsf{Eval}(\kappa, s)$.

We note that all known constructions of programmable hash functions [30, 27] are algebraic.

5.2 A Simple KEM Based on a Programmable Hash Function

We will now show how to construct an $\texttt{IND-}q\texttt{-CCA2}$ secure KEM with ciphertexts consisting of just a single group element, from an algebraic $(q, 1)$-programmable hash function.

Let $H = (\mathsf{HGen}, \mathsf{Eval}, \mathsf{HTrapGen}, \mathsf{HTrapEval})$ be an algebraic programmable hash function with respect to GS. Let $\ell(\lambda)$ be the input length of H. We also assume that any group element of $\mathbb{G}$ where $\Lambda = (g, p, \mathbb{G}) \leftarrow \mathsf{GS}(1^\lambda)$ can be described with $\ell(\lambda)$ bits. Using H as a building block, we construct a KEM Γ as follows:

[2] Note that $\mathsf{HTrapEval}$ is trivially an algebraic algorithm since it does not output any group elements of $\mathbb{G}$.

KG : On input $\Lambda = (g, p, \mathbb{G})$, pick randomness r_{HGen} for HGen and run $\kappa = (aux, X_1, \ldots, X_n) \leftarrow \mathsf{HGen}(\Lambda; r_{\mathsf{HGen}})$. Furthermore, run $\mathcal{E}_{\mathsf{HGen}}(\Lambda, r_{\mathsf{HGen}})$ to obtain $(y_1, \ldots, y_n)$ such that $X_i = g^{y_i}$ for all $i \in [n]$, and set $pk \leftarrow (\Lambda, \kappa)$ and $sk \leftarrow (\kappa, y_1, \ldots, y_n)$.

Enc : On input $pk = (\Lambda, \kappa)$, pick randomness $r \in \mathbb{Z}_p$, and compute the ciphertext $c = g^r$ and the session-key $K = \mathsf{Eval}(\kappa, c)^r$. (Here, c is treated as an $\ell(\lambda)$-bit string.)

Dec : On input $sk = (\kappa, y_1, \ldots, y_n)$ and $c = g^r$, compute $h \leftarrow \mathsf{Eval}(\kappa, c)$, run the extractor $\mathcal{E}_{\mathsf{Eval}}$ to obtain $(a_1, \ldots, a_n)$ satisfying $h = \prod_{i \in [n]} X_i^{a_i}$, and compute the session-key $K = c^{\sum_{i \in [n]} a_i y_i}$.

The correctness of the KEM follows from the properties of the extractors $\mathcal{E}_{\mathsf{HGen}}$ and $\mathcal{E}_{\mathsf{Eval}}$:

$$K = \mathsf{Eval}(\kappa, c)^r = h^r = (\prod_{i \in [n]} X_i^{a_i})^r = \prod_{i \in [n]} (g^r)^{a_i y_i} = c^{\sum_{i \in [n]} a_i y_i}$$

Note that the DDH-based KEM by Cramer et al. [13] can be seen as a concrete instantiation of the above KEM in which we use the concrete programmable hash function proposed in [27, Sect. 3.3].

Theorem 12 *Assume that H is an algebraic $(q, 1)$-programmable hash function with respect to* GS. *Then there exists an algebraic black-box reduction from the* `IND`-q-`CCA2` *security of the above KEM Γ to the hardness of the DDH problem with respect to* GS.

The proof of the above theorem is given in Appendix E. We note that using an almost identical proof, we can show that the above KEM is `OW`-q-`CCA2` secure under the hardness of the computational Diffie-Hellman (CDH) problem via an algebraic black-box reduction.

Note that due to the assumed algebraic property of the programmable hash function, the above KEM Γ falls into the class $\mathcal{K}_{\mathsf{GS},n}$ described in Section 3, where n is the number of group elements in the hash key of the programmable hash. Furthermore, the DDH problem is captured by the definition of a non-interactive problem described in Section 2.4. Hence Theorem 7 implies that there exists no algebraic black-box reduction from the `OW`-n-`CCA1` security of the KEM Γ to the hardness of any non-interactive problem[3]. On the other hand, the above Theorem 12 shows that such a reduction (from the stronger security notion `IND`-n-`CCA2`) is possible assuming the existence of an algebraic $(n, 1)$-programmable hash function. Since any (n, β)-programmable hash function is (n, β')-programmable if $\beta \geq \beta'$, this immediately gives us the following theorem.

Theorem 13 *For any group scheme* GS *and any integer $\beta \in \mathbb{N}$, there exists no algebraic (n, β)-programmable hash function with respect to* GS *whose hash key contains less than n group elements of $\mathbb{G}$, where $\mathbb{G}$ is the group described by Λ which is output by* GS.

Considering the case in which the parameter n for the programmable hash functions is considered to be any polynomial in λ, we obtain the following theorem, which answers the open question posed by Hofheinz and Kiltz [30] in the case of algebraic programmable hash function defined in prime order groups.

Theorem 14 *For any group scheme* GS *and any integer $\beta \in \mathbb{N}$, there exists no algebraic (poly, β)-programmable hash functions with respect to* GS.

6 Discussion

We have shown that there exists no algebraic black-box reduction from the `OW`-n-`CCA1` security of a KEM in the class $\mathcal{K}_{\mathsf{GS},n}$, to the hardness of any non-interactive problem with respect to GS. The class $\mathcal{K}_{\mathsf{GS},n}$ essentially captures the structure of the efficient KEMs [14, 28], [22, Sect. 4.1], and [34] (with explicit rejection), but requires the ciphertext to consist of just a single random group element and a string.

[3] Note that this does not contradict the results by Cramer et al. [13]. More specifically, while the KEM defined in [13] was shown to be `IND`-n-`CCA` under the DDH assumption via an algebraic black-box reduction, the scheme requires a public key containing $\mathcal{O}(n^2\lambda)$ group elements.

Our results leave several open problems. Specifically, it remains an open problem to prove the (non-)existence of a CCA secure KEM based on a non-interactive assumption, defined in a standard prime order group, and with a ciphertext overhead of just two group elements. Another interesting question is whether our results can be extended to rule out constrained CCA [28] secure KEMs based on non-interactive assumptions.

Furthermore, we have focused on simple KEMs defined in standard prime order groups, in which the session-key lies within the group. More precisely, the KEM class $\mathcal{K}_{\mathsf{GS},n}$ does not capture the structure of schemes which make use of a pairing to derive the session-key like [8], or apply a type of key-derivation function, such as the hardcore bit-based schemes like [22, Sect. 5], [25, Sect. 3], and [42, Sect. 3], the HDH-based versions of [22, Sect. 4.2] and [10, Sect. 6.2], or a combination of these like [25, Sect. 5.2 and 5.3] and [42, Sect. 5]. Note, however, that these schemes apply the key-derivation function (and/or the pairing) to one or more "seed" group elements to obtain a session-key. Here, it might be interesting to investigate the security provided by the "core part" of the schemes, in which the seed group element(s) is considered to be the session-key. Note that for all of the above mentioned schemes, these "core parts" can be shown to be $\mathtt{OW\text{-}CCA2}$ secure under an appropriate non-interactive assumption. Furthermore, the structure of these "core parts" is captured by $\mathcal{K}_{\mathsf{GS},n}$ if the ciphertext is reduced to consist of a single random group element and a string, for example, by applying the approach of compressing the group elements used for validity checking. In this case, our results imply that these "core parts" cannot be shown $\mathtt{OW\text{-}}n\mathtt{\text{-}CCA1}$ secure based on a non-interactive problem via an algebraic black-box reduction. This observation might provide some insight into the (im)possibility of constructing more efficient KEMs that make use of key-derivation functions, but drawing any formal conclusions regarding this, remains an open problem.

Since our results are restricted to KEMs defined in prime order groups, it is natural to ask whether similar results will hold in composite order groups. Note, however, that in composite order groups, it is possible to achieve a KEM with a ciphertext overhead of just a single group element [29, Sect. 5]. While this KEM only achieves constrained CCA security (based on a non-interactive assumption), it can be converted to a fully CCA secure KEM using the techniques from [2, 24] which will result in an additional ciphertext overhead of a MAC.

Lastly, we have shown lower bounds on the programmability of algebraic programmable hash functions in prime order groups. Note that all known constructions of programmable hash functions are algebraic [30, 27]. Furthermore, the definition of a programmable hash function requires the hash function to have some "algebraic properties" (see also the discussion in [30, Sect. 1.5]), which seems to suggest that standard model constructions of programmable hash functions are inherently algebraic.

Acknowledgement

We would like to thank Shota Yamada for suggesting looking at KEMs based on programmable hash functions, Masayuki Abe for sending a preliminary version of [1], and the anonymous reviewers for helpful comments. The second and third authors are support by JSPS Research Fellowships for Young Scientists.

References

1. M. Abe, J. Groth, and M. Ohkubo. Separating short structure-preserving signatures from non-interactive assumptions. In *ASIACRYPT*, pages 628–646, 2011.
2. J. Baek, D. Galindo, W. Susilo, and J. Zhou. Constructing strong kem from weak kem (or how to revive the kem/dem framework). In *SCN*, pages 358–374, 2008.
3. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *ASIACRYPT*, pages 531–545, 2000.
4. M. Bellare and A. Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In *CRYPTO*, pages 519–536, 1999.

5. D. Boneh and X. Boyen. Short signatures without random oracles. In *EUROCRYPT*, pages 56–73, 2004.
6. D. Boneh, P. A. Papakonstantinou, C. Rackoff, Y. Vahlis, and B. Waters. On the impossibility of basing identity based encryption on trapdoor permutations. In *FOCS*, pages 283–292, 2008.
7. D. Boneh and R. Venkatesan. Breaking rsa may not be equivalent to factoring. In *EUROCRYPT*, pages 59–71, 1998.
8. X. Boyen, Q. Mei, and B. Waters. Direct chosen ciphertext security from identity-based techniques. Cryptology ePrint Archive, Report 2005/288, 2005.
9. J. Camenisch, K. Haralambiev, M. Kohlweiss, J. Lapon, and V. Naessens. Structure preserving cca secure encryption and applications. In *ASIACRYPT*, pages 89–106, 2011.
10. D. Cash, E. Kiltz, and V. Shoup. The twin diffie-hellman problem and applications. Cryptology ePrint Archive, Report 2008/067, 2008. `http://eprint.iacr.org/`. This is the full version of [11].
11. D. Cash, E. Kiltz, and V. Shoup. The twin diffie-hellman problem and applications. In *EUROCRYPT*, pages 127–145, 2008.
12. J.-S. Coron. Optimal security proofs for pss and other signature schemes. In *EUROCRYPT*, pages 272–287, 2002.
13. R. Cramer, G. Hanaoka, D. Hofheinz, H. Imai, E. Kiltz, R. Pass, A. Shelat, and V. Vaikuntanathan. Bounded cca2-secure encryption. In *ASIACRYPT*, pages 502–518, 2007.
14. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Computing*, 33(1):167–226, 2003.
15. K. Emura, G. Hanaoka, T. Matsuda, G. Ohtake, and S. Yamada. Keyed-homomorphic public-key encryption: How to simultaneously achieve cca security and homomorphic operation, 2012. Manuscript.
16. M. Fischlin and D. Schröder. On the impossibility of three-move blind signature schemes. In *EUROCRYPT*, pages 197–215, 2010.
17. C. Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.
18. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108, 2011.
19. Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In *FOCS*, pages 325–335, 2000.
20. Y. Gertner, T. Malkin, and S. Myers. Towards a separation of semantic and cca security for public key encryption. In *TCC*, pages 434–455, 2007.
21. Y. Gertner, T. Malkin, and O. Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *FOCS*, pages 126–135, 2001.
22. G. Hanaoka and K. Kurosawa. Efficient chosen ciphertext secure public key encryption under the computational diffie-hellman assumption. Cryptology ePrint Archive, Report 2008/211, 2008. `http://eprint.iacr.org/`. This is the full version of [23].
23. G. Hanaoka and K. Kurosawa. Efficient chosen ciphertext secure public key encryption under the computational diffie-hellman assumption. In *ASIACRYPT*, pages 308–325, 2008.
24. G. Hanaoka and K. Kurosawa. Between hashed dh and computational dh: Compact encryption from weaker assumption. *IEICE Transactions*, 93-A(11):1994–2006, 2010.
25. K. Haralambiev, T. Jager, E. Kiltz, and V. Shoup. Simple and efficient public-key encryption from computational diffie-hellman in the standard model. In *Public Key Cryptography*, pages 1–18, 2010.
26. J. Herranz, D. Hofheinz, and E. Kiltz. Some (in)sufficient conditions for secure hybrid encryption. *Inf. Comput.*, 208(11):1243–1257, 2010.
27. D. Hofheinz, T. Jager, and E. Kiltz. Short signatures from weaker assumptions. In *ASIACRYPT*, pages 647–666, 2011.
28. D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO*, pages 553–571, 2007.
29. D. Hofheinz and E. Kiltz. The group of signed quadratic residues and applications. In *CRYPTO*, pages 637–653, 2009.
30. D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. *J. Cryptology*, 25(3):484–527, 2012.
31. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *STOC*, pages 44–61, 1989.
32. J. Katz and A. Yerukhimovich. On black-box constructions of predicate encryption from trapdoor permutations. In *ASIACRYPT*, pages 197–213, 2009.
33. E. Kiltz. Chosen-ciphertext secure key-encapsulation based on gap hashed diffie-hellman. In *Public Key Cryptography*, pages 282–297, 2007.

34. K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In *CRYPTO*, pages 426–442, 2004.
35. T. Matsuda and K. Matsuura. Parallel decryption queries in bounded chosen ciphertext attacks. In *Public Key Cryptography*, pages 246–264, 2011.
36. P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *ASIACRYPT*, pages 1–20, 2005.
37. R. Pass. Limits of provable security from standard assumptions. In *STOC*, pages 109–118, 2011.
38. D. H. Phan and D. Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). In *Selected Areas in Cryptography*, pages 182–197, 2004.
39. O. Reingold, L. Trevisan, and S. P. Vadhan. Notions of reducibility between cryptographic primitives. In *TCC*, pages 1–20, 2004.
40. D. R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *EUROCRYPT*, pages 334–345, 1998.
41. B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT 2005*, pages 114–127, 2005.
42. S. Yamada, Y. Kawai, G. Hanaoka, and N. Kunihiro. Public key encryption schemes from the (b)cdh assumption with better efficiency. *IEICE Transactions*, 93-A(11):1984–1993, 2010.
43. A. Yerukhimovich. A study of separation in cryptography: New results and new models, 2011. PhD thesis, the University of Maryland. Available at http://www.cs.umd.edu/˜arkady/thesis/thesis.pdf.

A Omitted Definitions and Descriptions

A.1 Basic Facts

In this paper, we will use the following facts.

Lemma 15 *(The Hoeffding bound) Let* $X_1, \dots, X_t$ *be independent random variables between* 0 *and* 1*, so that* $\Pr[X_i = 1] = \mu$ *for all* $i \in [t]$*. Let* $\chi = \frac{1}{t}\sum_{i \in [t]} X_i$*. Then, for any* $0 < \nu < 1$*,* $\Pr[\chi < (1 - \nu)\mu] < e^{-\nu^2 \mu t/2}$*.*

Lemma 16 *(Linear dependency) Let* $n = n(\lambda)$ *be a positive polynomial, and* p *be a prime such that* $|p|$ *is polynomial in* λ*. Furthermore, let* $\mathbf{a}_1, \dots, \mathbf{a}_{n-1}, \mathbf{b} \in \mathbb{Z}_p^n$ *be* n*-dimensional vectors. Then there is a deterministic polynomial time algorithm (in* λ*) that takes as input* $\mathbf{a}_1, \dots, \mathbf{a}_{n-1}, \mathbf{b}$*, and returns coefficients* $(\alpha_1, \dots, \alpha_{n-1}) \in \mathbb{Z}_p^{n-1}$ *satisfying* $\mathbf{b} = \sum_{i \in [n-1]} \alpha_i \mathbf{a}_i$ *if such coefficients exist, and otherwise returns* $\perp$*.*

Note that the algorithm mentioned in Lemma 16 could be implemented by performing Gauss-Jordan elimination on the matrix obtained by letting the vectors $\mathbf{a}_1, \dots, \mathbf{a}_{n-1}, \mathbf{b}$ correspond to the columns the matrix.

A.2 Key Encapsulation

Here, we review the definitions for a key encapsulation mechanism (KEM). Since in this paper we will only treat KEMs based on a group with prime order, for convenience we define a KEM with respect to a group scheme GS.

A KEM $\mathit{\Gamma}$ with respect to a group scheme GS consists of the following three PPTAs $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$:

KG: (Key Generation) This algorithm takes a group description $\mathit{\Lambda} = (g, p, \mathbb{G})$ (output from $\mathsf{GS}(1^\lambda)$) as input, and outputs a public/secret key pair (pk, sk). (Without loss of generality, we assume that pk and sk contain the information on $\mathit{\Lambda}$.)

Enc: (Encapsulation) This algorithm takes pk as input, and outputs a ciphertext c and a session-key $K \in \mathcal{K}$ (where $\mathcal{K}$ is a session-key space specified by pk).

Dec: (Decapsulation) This algorithm takes sk and c as input, and outputs a session-key K which could be a special symbol $\perp$ meaning "invalid".

We require $\mathsf{Dec}(sk, c) = K$ for all (pk, sk) output by $\mathsf{KG}(\mathit{\Lambda})$ and all (c, K) output by $\mathsf{Enc}(pk)$.

```
$\mathsf{Expt}^{\mathtt{IND\text{-}ATK}}_{\Gamma,\mathcal{A}}(\lambda)$ :
    $\Lambda \leftarrow \mathsf{GS}(1^\lambda)$;
    $(pk, sk) \leftarrow \mathsf{KG}(\Lambda)$;
    $\mathsf{st} \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$;
    $(c^*, K_1^*) \leftarrow \mathsf{Enc}(pk)$;
    $K_0^* \leftarrow \mathcal{K}$;
    $b \leftarrow \{0,1\}$;
    $b' \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(\mathsf{st}, c^*, K_b^*)$;
    Return $(b' \stackrel{?}{=} b)$
```

```
$\mathsf{Expt}^{\mathtt{OW\text{-}ATK}}_{\Gamma,\mathcal{A}}(\lambda)$:
    $\Lambda \leftarrow \mathsf{GS}(1^\lambda)$;
    $(pk, sk) \leftarrow \mathsf{KG}(\Lambda)$;
    $\mathsf{st} \leftarrow \mathcal{A}_1^{\mathcal{O}_1}(pk)$;
    $(c^*, K^*) \leftarrow \mathsf{Enc}(pk)$;
    $K' \leftarrow \mathcal{A}_2^{\mathcal{O}_2}(\mathsf{st}, c^*)$;
    Return $(K' \stackrel{?}{=} K^*)$
```

Definitions of Oracles

ATK	$\mathcal{O}_1(\cdot)$	$\mathcal{O}_2(\cdot)$
CPA	$\perp$	$\perp$
q-CCA1, CCA1	$\mathsf{Dec}(sk, \cdot)$	$\perp$
q-CCA2, CCA2	$\mathsf{Dec}(sk, \cdot)$	$\mathsf{Dec}(sk, \cdot)^{(\dagger)}$

$^{(\dagger)}$ c^* cannot be submitted.

Fig. 1. The security experiments for a KEM (with respect to a group scheme GS): The experiment for indistinguishability (IND-ATK security) (left), that for one-wayness (OW-ATK security) (center), and the definitions of oracles (right).

Security Notions for KEMs. Typically, security notions for KEMs are expressed by the combination of a security goal (GOAL) and an adversary's attack type (ATK). In this paper, we will treat *indistinguishability* (IND) and *one-wayness* (OW) as security goals GOAL, and *chosen plaintext attacks* (CPA), *non-adaptive chosen ciphertext attacks* (CCA1), *adaptive chosen ciphertext attacks* (CCA2), and their q-bounded analogues [13] (q-CCA1 and q-CCA2) as an adversary's attack types ATK.

For a KEM $\Gamma = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ with respect to GS, we define the experiment $\mathsf{Expt}^{\mathtt{IND\text{-}ATK}}_{\Gamma,\mathcal{A}}(\lambda)$ in which an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ attacks the indistinguishability of Γ under the attack type ATK, and the experiment $\mathsf{Expt}^{\mathtt{OW\text{-}ATK}}_{\Gamma,\mathcal{A}}(\lambda)$ in which $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ attacks the one-wayness of Γ under ATK, as shown in Fig. 1. In the experiments, if $\mathtt{ATK} \in \{q\text{-}\mathtt{CCA1}, q\text{-}\mathtt{CCA2}\}$, then $\mathcal{A}$ is allowed to use $\mathcal{O}_1$ and $\mathcal{O}_2$ at most q times in total.

For a KEM Γ, an adversary $\mathcal{A}$, and $\mathtt{ATK} \in \{\mathtt{CPA}, q\text{-}\mathtt{CCA1}, \mathtt{CCA1}, q\text{-}\mathtt{CCA2}, \mathtt{CCA2}\}$ (with $q \in \mathbb{N}$), we define $\mathcal{A}$'s IND-ATK advantage $\mathsf{Adv}^{\mathtt{IND\text{-}ATK}}_{\Gamma,\mathcal{A}}(\lambda)$ and OW-ATK advantage $\mathsf{Adv}^{\mathtt{OW\text{-}ATK}}_{\Gamma,\mathcal{A}}(\lambda)$ as follows:

$$\mathsf{Adv}^{\mathtt{IND\text{-}ATK}}_{\Gamma,\mathcal{A}}(\lambda) = \Pr[\mathsf{Expt}^{\mathtt{IND\text{-}ATK}}_{\Gamma,\mathcal{A}}(\lambda) = 1] - \frac{1}{2}$$
$$\mathsf{Adv}^{\mathtt{OW\text{-}ATK}}_{\Gamma,\mathcal{A}}(\lambda) = \Pr[\mathsf{Expt}^{\mathtt{OW\text{-}ATK}}_{\Gamma,\mathcal{A}}(\lambda) = 1]$$

Definition 17 *Let* $\mathtt{GOAL} \in \{\mathtt{IND}, \mathtt{OW}\}$ *and* $\mathtt{ATK} \in \{\mathtt{CPA}, q\text{-}\mathtt{CCA1}, \mathtt{CCA1}, q\text{-}\mathtt{CCA2}, \mathtt{CCA2}\}$ *(with* $q \in \mathbb{N}$*). We say that a KEM* Γ *is* GOAL-ATK *secure if for any PPTA* $\mathcal{A}$*, there exists a negligible function* $\mu(\cdot)$ *such that* $\mathsf{Adv}^{\mathtt{GOAL\text{-}ATK}}_{\Gamma,\mathcal{A}}(\lambda) \leq \mu(\lambda)$*.*

A.3 Concrete Non-interactive Problems

In this paper, we make use of the (hardness of the) following concrete non-interactive problems. Although these problems are special cases of a NIP, it is convenient for us to define them separately, and thus we review them below for completeness. As usual, the following problems are said to be hard if the advantage functions are negligible in the security parameter λ for any PPTA.

Discrete Logarithm Problem. Let GS be a group scheme and let $\mathcal{A}$ be an algorithm. We define the advantage $\mathsf{Adv}^{\mathtt{DL}}_{\mathsf{GS},\mathcal{A}}(\lambda)$ of $\mathcal{A}$ in solving the discrete logarithm (DL) problem with respect to GS as follows:

$$\mathsf{Adv}^{\mathtt{DL}}_{\mathsf{GS},\mathcal{A}}(\lambda) = \Pr[\Lambda = (g, p, \mathbb{G}) \leftarrow \mathsf{GS}(1^\lambda);\ \alpha \leftarrow \mathbb{Z}_p : \mathcal{A}(\Lambda, g^\alpha) = \alpha]$$

Decisional Diffie-Hellman Problem. Let GS be a group scheme and let $\mathcal{A}$ be an algorithm. We define the advantage $\mathsf{Adv}^{\mathtt{DDH}}_{\mathsf{GS},\mathcal{A}}(\lambda)$ of $\mathcal{A}$ in solving the decisional Diffie-Hellman (DDH) problem with respect to GS as follows:

$$\mathsf{Adv}^{\mathtt{DDH}}_{\mathsf{GS},\mathcal{A}}(\lambda) = \Pr[\Lambda = (g, p, \mathbb{G}) \leftarrow \mathsf{GS}(1^\lambda);\ \alpha, \beta, \gamma \leftarrow \mathbb{Z}_p;$$
$$b \leftarrow \{0,1\};\ Z_0 \leftarrow g^\gamma;\ Z_1 \leftarrow g^{\alpha\beta} : \mathcal{A}(\Lambda, g^\alpha, g^\beta, Z_b) = b] - \frac{1}{2}$$

B Proof of Lemma 8

Proof. Let GS, Γ, and $\mathsf{P} = (\mathsf{I}, \mathsf{V}, \mathsf{U})$ be as stated in the lemma. Fix $\mathtt{GOAL} \in \{\mathtt{OW}, \mathtt{IND}\}$ and $\mathtt{ATK} \in \{\mathtt{CPA}, q\text{-}\mathtt{CCA1}, \mathtt{CCA1}, q\text{-}\mathtt{CCA2}, \mathtt{CCA2}\}$ (with $q \in \mathbb{N}$). Towards a contradiction, assume simultaneously that P is hard and that there exists an algebraic black-box reduction from the $\mathtt{GOAL\text{-}ATK}$ security of Γ to the hardness of P. The latter guarantees that there exists an algebraic oracle PPTA $\mathcal{R}$ that satisfies the condition of Definition 5.

We first consider the case in which the DL problem with respect to GS is not hard. That is, we assume there exists a PPTA algorithm $\mathcal{A}$ such that $\mathsf{Adv}^{\mathtt{DL}}_{\mathsf{GS},\mathcal{A}}(\lambda)$ is non-negligible. Consider the following (trivial) adversary $\mathcal{A}'$ that uses $\mathcal{A}$ as a subroutine and attacks the $\mathtt{OW\text{-}CPA}$ security of Γ:

On input a public key $pk = (\Lambda, aux, X_1, \dots, X_n)$ (where $\Lambda = (g, p, \mathbb{G})$) and the challenge ciphertext $C^* = (c^*, d^*) \in \mathbb{G} \times \{0,1\}^*$, $\mathcal{A}'$ runs $\mathcal{A}$ with input (Λ, c^*). When $\mathcal{A}$ terminates with output r, $\mathcal{A}'$ returns the session-key $K = g^{f_0(pk,r)} \prod_{i \in [n]} X_i^{f_i(pk,r)}$, where $\{f_i\}_{i \in \{0,\dots,n\}}$ are the scheme-dependent functions defined by Γ.

It is easy to see that $\mathcal{A}'$ succeeds in breaking the $\mathtt{OW\text{-}CPA}$ security of Γ whenever $\mathcal{A}$ succeeds in calculating the discrete logarithm $r = \log_g c^*$, and thus $\mathsf{Adv}^{\mathtt{OW\text{-}CPA}}_{\Gamma,\mathcal{A}'}(\lambda) = \mathsf{Adv}^{\mathtt{DL}}_{\mathsf{GS},\mathcal{A}}(\lambda)$ is non-negligible. Then, since any $\mathtt{GOAL\text{-}ATK}$ security considered in this lemma implies $\mathtt{OW\text{-}CPA}$ security, $\mathcal{A}'$ can be easily modified so that it has non-negligible advantage in breaking $\mathtt{GOAL\text{-}ATK}$ security of Γ. Hence, by the definition of an algebraic black-box reduction, $\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{R}^{\mathcal{A}'}}(\lambda)$ must also be non-negligible. Since both $\mathcal{R}$ and $\mathcal{A}'$ are PPTAs, $\mathcal{R}^{\mathcal{A}'}$ can be rewritten as a single PPTA $\mathcal{R}'$ which will have non-negligible advantage in solving P. However, this contradicts the fact that P is a hard NIP, and thereby the existence of $\mathcal{R}$.

We will now consider the case in which the DL problem with respect to GS is hard. The proof makes use of the distribution $\mathbb{D}$ of the oracles $\mathcal{O}$ provided by the assumption in the lemma. More specifically, by the first property of $\mathbb{D}$, there exists an algebraic oracle PPTA $\mathcal{A}$ and some positive polynomial $p(\lambda)$ such that for infinitely many λ's,

$$\mathop{\mathbf{E}}_{\mathcal{O} \leftarrow \mathbb{D}}\left[\mathsf{Adv}^{\mathtt{GOAL\text{-}ATK}}_{\Gamma,\mathcal{A}^{\mathcal{O}}}(\lambda)\right] = \mathop{\mathbf{E}}_{\mathcal{O} \leftarrow \mathbb{D}}\left[\Pr[\mathsf{Expt}^{\mathtt{GOAL\text{-}ATK}}_{\Gamma,\mathcal{A}^{\mathcal{O}}}(\lambda) = 1] - \rho\right] \geq \frac{1}{p(\lambda)},$$

where $\rho = 0$ if $\mathtt{GOAL} = \mathtt{OW}$ and $\rho = \frac{1}{2}$ if $\mathtt{GOAL} = \mathtt{IND}$. Then, by a simple averaging argument, we have (for infinitely many λ's):

$$\Pr_{\mathcal{O} \leftarrow \mathbb{D}}\left[\mathsf{Adv}^{\mathtt{GOAL\text{-}ATK}}_{\Gamma,\mathcal{A}^{\mathcal{O}}}(\lambda) \geq \frac{1}{2p(\lambda)}\right] > \frac{1}{2p(\lambda)}$$

Call $\mathcal{O} \in [\mathbb{D}]$ *good* if $\mathsf{Adv}^{\mathtt{GOAL\text{-}ATK}}_{\Gamma,\mathcal{A}^{\mathcal{O}}}(\lambda) \geq \frac{1}{2p(\lambda)}$. Then, by definition we have

$$\Pr_{\mathcal{O} \leftarrow \mathbb{D}}[\mathcal{O} \text{ is good}] > \frac{1}{2p(\lambda)} \tag{2}$$

Define the algebraic oracle PPTA $\widetilde{\mathcal{R}}^{\mathcal{O}}$ by $\mathcal{R}^{\mathcal{A}^{\mathcal{O}}}$. Note that such algebraic oracle PPTA $\widetilde{\mathcal{R}}$ must exist because $\mathcal{R}$ and $\mathcal{A}$ are both algebraic oracle PPTAs. The corresponding extractor for $\widetilde{\mathcal{R}}$ can be easily constructed by appropriately combining the extractors of $\mathcal{R}$ and $\mathcal{A}$.

Then, we know that under a good oracle $\widehat{\mathcal{O}}$, $\mathcal{A}^{\widehat{\mathcal{O}}}$ has non-negligible advantage (at least $\frac{1}{2p(\lambda)}$) in breaking the $\mathtt{GOAL\text{-}ATK}$ security of Γ. Therefore, by definition of an algebraic black-box reduction, $\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\widetilde{\mathcal{R}}^{\widehat{\mathcal{O}}}}(\lambda) = \mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{R}^{\mathcal{A}^{\widehat{\mathcal{O}}}}}(\lambda)$ is non-negligible. That is, there exists another positive polynomial $p'(\lambda)$ such that, for infinitely many λ's, we have

$$\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\widetilde{\mathcal{R}}^{\widehat{\mathcal{O}}}}(\lambda) = \Pr[\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\widetilde{\mathcal{R}}^{\widehat{\mathcal{O}}}}(\lambda) = 1] - \delta(\lambda) \geq \frac{1}{p'(\lambda)}$$

Here, let us introduce some notation for convenience. For an oracle algorithm $\mathcal{M}$ that has access to an oracle in $[\mathbb{D}]$, we denote by $\widetilde{\mathsf{Expt}}^{\mathsf{P}}_{\mathsf{GS},\mathcal{M}^{\mathbb{D}}}(\lambda)$ the following experiment:

$$\widetilde{\mathsf{Expt}}^{\mathsf{P}}_{\mathsf{GS},\mathcal{M}^{\mathbb{D}}}(\lambda) : [\mathcal{O} \leftarrow \mathbb{D}; \text{Return } \mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\mathcal{M}^{\mathcal{O}}}(\lambda)]$$

Note that by definition, for an oracle algorithm $\mathcal{M}$, it holds that

$$\mathop{\mathbf{E}}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{M}^{\mathcal{O}}}(\lambda)] = \mathop{\mathbf{E}}_{\mathcal{O}\leftarrow\mathbb{D}}\left[\Pr[\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\mathcal{M}^{\mathcal{O}}}(\lambda)=1]-\delta(\lambda)\right] = \Pr[\widetilde{\mathsf{Expt}}^{\mathsf{P}}_{\mathsf{GS},\mathcal{M}^{\mathbb{D}}}(\lambda)=1]-\delta(\lambda)$$

Now, in order to reach a contradiction with the assumption that P is hard (using the second property of $\mathbb{D}$ and the assumption that the DL problem is hard), we would like to show the existence of an algebraic oracle PPTA $\widehat{\mathcal{R}}$ such that the "expected advantage" $\mathbf{E}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\widehat{\mathcal{R}}^{\mathcal{O}}}(\lambda)]$ is non-negligible. One might think that the algorithm $\widetilde{\mathcal{R}}$ can be used for that purpose. However, $\widetilde{\mathcal{R}}$ is only guaranteed to have non-negligible advantage in solving P *when the given oracle* $\mathcal{O}$ *causes* $\mathcal{A}$ *to have non-negligible advantage in breaking the* `GOAL-ATK` *security of* Γ. When this is not the case, nothing is guaranteed about the success probability of $\widetilde{\mathcal{R}}$ which might even be negative. Note that this might cause the expected advantage $\mathbf{E}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\widetilde{\mathcal{R}}^{\mathcal{O}}}(\lambda)]$ to be not non-negligible.

Instead, we consider the following algebraic oracle algorithm $\widehat{\mathcal{R}}$ that uses $\widetilde{\mathcal{R}}$ and U as building blocks, and runs in the experiment $\widetilde{\mathsf{Expt}}^{\mathsf{P}}_{\mathsf{GS},\widehat{\mathcal{R}}^{\mathbb{D}}}(\lambda)$. Intuitively, $\widehat{\mathcal{R}}$ first tests whether the given oracle $\mathcal{O}$ is "good" by running $\widetilde{\mathcal{R}}$ many times with independently generated instances (using I) and checking whether $\widetilde{\mathcal{R}}$ succeeds in solving these with sufficiently high probability, and then decides which of $\widetilde{\mathcal{R}}$ or U it uses to find the answer x to the given instance y. Specifically, the description of $\widehat{\mathcal{R}}$ is as follows:

$\widehat{\mathcal{R}}^{\mathcal{O}}(1^{\lambda}, y)$**:** (where y contains the group description Λ) Let $t = 8\lambda p'(\lambda)^2$ and $Z = \delta(\lambda) + \frac{1}{2p'(\lambda)}$. For $i \in [t]$, $\widehat{\mathcal{R}}$ runs $(y_i, w_i) \leftarrow \mathsf{I}(\Lambda)$, $x_i \leftarrow \widetilde{\mathcal{R}}^{\mathcal{O}}(1^{\lambda}, y_i)$, and $\mathsf{V}(y_i, x_i, w_i)$. We let $b_i = 1$ if $\mathsf{V}(y_i, x_i, w_i) = \top$ and $b_i = 0$ otherwise. $\widehat{\mathcal{R}}$ calculates $\chi = \frac{1}{t}\sum_{i\in[t]} b_i$. If $\chi > Z$ then $\widehat{\mathcal{R}}$ runs $x \leftarrow \widetilde{\mathcal{R}}^{\mathcal{O}}(1^{\lambda}, y)$, and otherwise runs $x \leftarrow \mathsf{U}(1^{\lambda}, y)$. Finally, $\widehat{\mathcal{R}}$ terminates with output x.

It is easy to see that $\widehat{\mathcal{R}}$ is an algebraic oracle PPTA, because both $\widetilde{\mathcal{R}}$ and U are algebraic (oracle) PPTAs and $p'(\lambda)$ is a positive polynomial.

Call $\mathcal{O} \in [\mathbb{D}]$ *positive* if $\Pr[\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\widetilde{\mathcal{R}}^{\mathcal{O}}}(\lambda) = 1] - \delta(\lambda) \geq 0$, and otherwise call $\mathcal{O}$ *negative*. Note that by definition, any good oracle $\mathcal{O} \in [\mathbb{D}]$ is also positive.

We will now estimate $\mathbf{E}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\widehat{\mathcal{R}}^{\mathcal{O}}}(\lambda)] = \Pr[\widetilde{\mathsf{Expt}}^{\mathsf{P}}_{\mathsf{GS},\widehat{\mathcal{R}}^{\mathbb{D}}}(\lambda) = 1] - \delta(\lambda)$. In the experiment $\widetilde{\mathsf{Expt}}^{\mathsf{P}}_{\mathsf{GS},\widehat{\mathcal{R}}^{\mathbb{D}}}(\lambda)$, we consider the following events:

- Succ: $\widehat{\mathcal{R}}$ succeeds in outputting x such that $\mathsf{V}(y, x, w) = \top$.
- Posi: $\mathcal{O}$ (chosen according to $\mathbb{D}$) is positive
- Good: $\mathcal{O}$ (chosen according to $\mathbb{D}$) is good.

Here, according to our description of $\widehat{\mathcal{R}}$, we know that the following two equations hold (where Λ is the group description output from $\mathsf{GS}(1^{\lambda})$)[4]:

$$\begin{aligned}&\Pr[\mathsf{Succ}|\chi > Z \wedge \mathsf{Good}] - \delta(\lambda)\\ &\qquad = \Pr[(y,w)\leftarrow \mathsf{I}(\Lambda);\ x \leftarrow \widetilde{\mathcal{R}}^{\mathcal{O}}(1^{\lambda}, y) : \mathsf{V}(y,x,w) = \top | \mathcal{O} \text{ is good}] - \delta(\lambda) \geq \frac{1}{p'(\lambda)}\end{aligned} \quad (3)$$

$$\Pr[\mathsf{Succ}|\chi \leq Z] = \Pr[(y,w)\leftarrow \mathsf{I}(\Lambda);\ x \leftarrow \mathsf{U}(1^{\lambda}, y) : \mathsf{V}(y,x,w) = \top] = \delta(\lambda) \quad (4)$$

[4] Recall that we define a group scheme GS so that a group description is deterministically determined by the security parameter λ, and thus the success probability of an adversary solving a NIP P is not affected by the generation of Λ for a fixed security parameter λ.

Using these, we have (for infinitely many λ's):

$$\begin{aligned}
&\mathop{\mathbf{E}}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\widehat{\mathcal{R}}^{\mathcal{O}}}(\lambda)] = \Pr[\widetilde{\mathsf{Expt}}^{\mathsf{P}}_{\mathsf{GS},\widehat{\mathcal{R}}^{\mathbb{D}}}(\lambda)=1] - \delta(\lambda) = \Pr[\mathsf{Succ}] - \delta(\lambda)\\
&= \Pr[\mathsf{Succ}\wedge\chi > Z] + \Pr[\mathsf{Succ}|\chi \le Z]\cdot\Pr[\chi\le Z] - \delta(\lambda)\\
&\overset{(*)}{=} \Pr[\mathsf{Succ}\wedge\chi > Z] + \delta(\lambda)\cdot(1-\Pr[\chi > Z]) - \delta(\lambda)\\
&= \Pr[\mathsf{Succ}\wedge\chi > Z] - \delta(\lambda)\cdot\Pr[\chi > Z]\\
&= \Pr[\mathsf{Succ}\wedge\chi > Z] - \delta(\lambda)\cdot(\Pr[\chi > Z\wedge\mathsf{Posi}] + \Pr[\chi > Z|\overline{\mathsf{Posi}}]\cdot\Pr[\overline{\mathsf{Posi}}])\\
&\ge \Pr[\mathsf{Succ}\wedge\chi > Z\wedge\mathsf{Posi}] - \delta(\lambda)\cdot\Pr[\chi > Z\wedge\mathsf{Posi}] - \Pr[\chi > Z|\overline{\mathsf{Posi}}]\\
&= \Pr[\mathsf{Succ}\wedge\chi > Z\wedge\mathsf{Good}] + \Pr[\mathsf{Succ}\wedge\chi > Z\wedge\mathsf{Posi}\wedge\overline{\mathsf{Good}}]\\
&\quad - \delta(\lambda)\cdot(\Pr[\chi > Z\wedge\mathsf{Good}] + \Pr[\chi > Z\wedge\mathsf{Posi}\wedge\overline{\mathsf{Good}}]) - \Pr[\chi > Z|\overline{\mathsf{Posi}}]\\
&= \Pr[\chi > Z\wedge\mathsf{Good}]\cdot(\Pr[\mathsf{Succ}|\chi > Z\wedge\mathsf{Good}] - \delta(\lambda))\\
&\quad + \Pr[\chi > Z\wedge\mathsf{Posi}\wedge\overline{\mathsf{Good}}]\cdot(\Pr[\mathsf{Succ}|\chi > Z\wedge\mathsf{Posi}\wedge\overline{\mathsf{Good}}] - \delta(\lambda)) - \Pr[\chi > Z|\overline{\mathsf{Posi}}]\\
&\ge \Pr[\chi > Z|\mathsf{Good}]\cdot\Pr[\mathsf{Good}]\cdot(\Pr[\mathsf{Succ}|\chi > Z\wedge\mathsf{Good}] - \delta(\lambda)) - \Pr[\chi > Z|\overline{\mathsf{Posi}}]\\
&> \Pr[\chi > Z|\mathsf{Good}]\cdot\frac{1}{2p(\lambda)}\cdot\frac{1}{p'(\lambda)} - \Pr[\chi > Z|\overline{\mathsf{Posi}}]
\end{aligned}$$

where, in the above, the equation (*) is due to the equation (4), the second last inequality uses $\Pr[\mathsf{Succ}|\chi > Z\wedge\mathsf{Posi}\wedge\overline{\mathsf{Good}}] - \delta(\lambda) \ge 0$ which is because of the definition of the positive oracles, and the last inequality holds due to the equations (2) and (3) for infinitely many λ's.

Here, in order to proceed, we need the following claims, which will be proven later:

Claim 18 $\Pr[\chi > Z|\mathsf{Good}] > 1 - e^{-\lambda}$.

Claim 19 $\Pr[\chi > Z|\overline{\mathsf{Posi}}] < e^{-\lambda}$.

Using these, we have

$$\begin{aligned}
\mathop{\mathbf{E}}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\widehat{\mathcal{R}}^{\mathcal{O}}}(\lambda)] &> (1-e^{-\lambda})\cdot\frac{1}{2p(\lambda)}\cdot\frac{1}{p'(\lambda)} - e^{-\lambda}\\
&\ge \frac{1}{2p(\lambda)p'(\lambda)} - 2e^{-\lambda}
\end{aligned}$$

where the last inequality holds for infinitely many λ's. Put differently, the expected advantage of the algebraic oracle algorithm $\widehat{\mathcal{R}}$ in solving P is non-negligible.

However, by the second property of $\mathbb{D}$, there exist PPTAs $\mathcal{B}_1$ and $\mathcal{B}_2$, a polynomial $Q(\lambda)$, and a negligible function $\mu(\lambda)$ such that

$$\mathop{\mathbf{E}}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\widehat{\mathcal{R}}^{\mathcal{O}}}(\lambda)] \le Q(\lambda)\cdot\mathsf{Adv}^{\mathsf{DL}}_{\mathsf{GS},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{B}_2}(\lambda) + \mu(\lambda)$$

and since we are assuming that the DL problem is hard, $\mathsf{Adv}^{\mathsf{DL}}_{\mathsf{GS},\mathcal{B}_1}(\lambda)$ is negligible. Hence, it follows that $\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{B}_2}(\lambda) \ge \mathbf{E}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{R}'^{\mathcal{O}}}(\lambda)] - Q(\lambda)\cdot\mathsf{Adv}^{\mathsf{DL}}_{\mathsf{GS},\mathcal{B}_1}(\lambda) - \mu(\lambda)$ is non-negligible. However, this contradicts that the NIP P is hard, and thereby the existence of the algebraic black-box reduction $\mathcal{R}$.

It remains to prove Claims 18 and 19. For notational convenience, for an oracle $\mathcal{O}\in[\mathbb{D}]$ let

$$\begin{aligned}
P(\mathcal{O}) &= \Pr[\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\widetilde{\mathcal{R}}^{\mathcal{O}}}(\lambda)=1]\\
&= \Pr[(y,w)\leftarrow\mathsf{I}(\Lambda); x\leftarrow\widetilde{\mathcal{R}}^{\mathcal{O}}(1^\lambda,y) : \mathsf{V}(y,x,w)=\top]
\end{aligned}$$

where Λ is the group description output by $\mathsf{GS}(1^\lambda)$.

Proof. (of Claim 18.) Fix a good oracle $\mathcal{O}$. This guarantees that $P(\mathcal{O}) > \delta(\lambda) + \frac{1}{p'(\lambda)}$. For $i \in [t]$, let $\alpha_i \in \{0,1\}$ be the indicator variable which is set 1 if and only if $\mathsf{V}(y_i, x_i, w_i) = \top$. Note that $\mathbf{E}[\alpha_i] = \Pr[\alpha_i = 1] = P(\mathcal{O})$ for all $i \in [t]$. Using the Hoeffding bound (Lemma 15) with parameters $t = 8\lambda p'(\lambda)^2$, $Z = \delta(\lambda) + \frac{1}{2p'(\lambda)}$, $\mu = P(\mathcal{O})$, and $(1-\nu)\mu = Z$ (which implies $\nu = \frac{P(\mathcal{O})-Z}{P(\mathcal{O})}$), we can estimate the probability $\Pr[\chi \le Z|\mathsf{Good}]$ as follows:

$$\begin{aligned}
\Pr[\chi \le Z|\mathsf{Good}] &\le \exp(-\nu^2\mu t/2) \\
&= \exp\left(-\left(\frac{P(\mathcal{O})-Z}{P(\mathcal{O})}\right)^2 \cdot P(\mathcal{O}) \cdot (8\lambda p'(\lambda)^2)/2\right) \\
&= \exp\left(-\frac{1}{P(\mathcal{O})} \cdot (P(\mathcal{O})-Z)^2 \cdot 4\lambda p'(\lambda)^2\right) \\
&\overset{(*)}{<} \exp\left(-\left(\frac{1}{2p'(\lambda)}\right)^2 \cdot 4\lambda p'(\lambda)^2\right) \\
&= \exp(-\lambda)
\end{aligned}$$

where in the inequality (*) we used $\frac{1}{P(\mathcal{O})} > 1$ and $P(\mathcal{O}) - Z > \frac{1}{2p'(\lambda)}$. Therefore, we have $\Pr[\chi > Z|\mathsf{Good}] > 1 - e^{-\lambda}$. This completes the proof of Claim 18. □

Proof. (of Claim 19.) Fix a negative oracle $\mathcal{O}$. This guarantees that $P(\mathcal{O}) < \delta(\lambda)$. Let $Z' = 1 - Z$ and $\chi' = 1 - \chi$. Then the event $\chi > Z$ is equivalent to the event $\chi' < Z'$. For $i \in [t]$, let $\alpha'_i \in \{0,1\}$ be the indicator variable which is set to 1 if and only if $\mathsf{V}(y_i, x_i, w_i) = \bot$. Note that $\mathbf{E}[\alpha'_i] = \Pr[\alpha'_i = 1] = 1 - P(\mathcal{O})$ for all $i \in [t]$. Using the Hoeffding bound (Lemma 15) with parameters $t = 8\lambda p'(\lambda)^2$, $Z' = 1 - Z = 1 - (\delta(\lambda) + \frac{1}{2p'(\lambda)})$, $\mu = 1 - P(\mathcal{O})$, and $(1-\nu)\mu = Z'$ (which implies $\nu = \frac{Z-P(\mathcal{O})}{1-P(\mathcal{O})}$), we can estimate the probability $\Pr[\chi > Z|\overline{\mathsf{Posi}}]$ as follows:

$$\begin{aligned}
\Pr[\chi > Z|\overline{\mathsf{Posi}}] &= \Pr[\chi' < Z'|\overline{\mathsf{Posi}}] \\
&< \exp(-\nu^2\mu t/2) \\
&= \exp\left(-\left(\frac{Z-P(\mathcal{O})}{1-P(\mathcal{O})}\right)^2 \cdot (1-P(\mathcal{O})) \cdot (8\lambda p'(\lambda)^2)/2\right) \\
&= \exp\left(-\frac{1}{1-P(\mathcal{O})} \cdot (Z-P(\mathcal{O}))^2 \cdot 4\lambda p'(\lambda)^2\right) \\
&\overset{(*)}{<} \exp\left(-\left(\frac{1}{2p'(\lambda)}\right)^2 \cdot 4\lambda p'(\lambda)^2\right) \\
&= \exp(-\lambda),
\end{aligned}$$

where in the inequality (*) we used $\frac{1}{1-P(\mathcal{O})} > 1$ and $Z - P(\mathcal{O}) > \frac{1}{2p'(\lambda)}$. This completes the proof of Claim 19. □

We have seen that regardless of the hardness of the DL problem, if the NIP P is hard, then there is no algebraic black-box reduction from the `GOAL-ATK` security of a KEM $\Gamma \in \mathcal{K}_{\mathsf{GS},n}$ to the hardness of P. The above proof works for any group scheme GS, any KEM $\Gamma \in \mathcal{K}_{\mathsf{GS},n}$ and any NIP P with respect to GS. This completes the proof of Lemma 8. □

C Proof of Lemma 9

Proof. Let GS, Γ, and $\mathbb{D}$ be as stated in the lemma. For an algebraic oracle algorithm $\mathcal{A}$ that has access to an oracle $\mathcal{O} \in [\mathbb{D}]$, let us denote by $\widetilde{\mathsf{Expt}}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathbb{D}}}(\lambda)$ the experiment $\mathsf{Expt}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathcal{O}}}(\lambda)$ in which the choice of the oracle $\mathcal{O}$ according to the distribution $\mathbb{D}$ is also taken into account. That is,

$$\widetilde{\mathsf{Expt}}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathbb{D}}}(\lambda) : [\mathcal{O} \leftarrow \mathbb{D};\ \text{Return } \mathsf{Expt}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathcal{O}}}(\lambda)]$$

Then, by definition, we have

$$\mathop{\mathbf{E}}_{\mathcal{O}\leftarrow\mathbb{D}}\left[\mathsf{Adv}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathcal{O}}}(\lambda)\right] = \mathop{\mathbf{E}}_{\mathcal{O}\leftarrow\mathbb{D}}\left[\ \Pr[\mathsf{Expt}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathcal{O}}}(\lambda) = 1]\ \right] = \Pr[\widetilde{\mathsf{Expt}}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathbb{D}}}(\lambda) = 1]$$

Therefore, to prove the lemma, it is sufficient to show an algebraic oracle PPTA $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and lowerbound $\mathcal{A}$'s success probability in the experiment $\widetilde{\mathsf{Expt}}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathbb{D}}}(\lambda)$. As defined by the experiment $\widetilde{\mathsf{Expt}}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathbb{D}}}(\lambda)$, $\mathcal{A}_1$ is given a public key $pk = (\Lambda, aux, X_1, \dots, X_n)$ of Γ as input, and access to a corresponding decapsulation oracle $\mathcal{O}_{dec}$. Furthermore, both $\mathcal{A}_1$ and $\mathcal{A}_2$ are given access to the oracle $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$ which is chosen according to $\mathbb{D}$ in the experiment. $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is constructed as follows:

$\mathcal{A}_1^{\mathcal{O}_{dec},\mathcal{O}_1,\mathcal{O}_2}(pk = (\Lambda, aux, X_1, \dots, X_n))$**:** (Let $\Lambda = (g, p, \mathbb{G})$, and let $y_i = \log_g X_i$ for $i \in [n]$) $\mathcal{A}_1$ submits pk to $\mathcal{O}_1$ to obtain $(C_1, \dots, C_n, \sigma)$, and then submits C_i to $\mathcal{O}_{dec}$ to obtain K_i for each $i \in [n]$. $\mathcal{A}_1$ then submits $(pk, K_1, \dots, K_n, \sigma)$ to $\mathcal{O}_2$ to obtain $\{u_i = \psi_1(pk, C_i, y_1, \dots, y_n)\}_{i\in[n]}$. Note that in the experiment, since pk is generated correctly and $\mathcal{O}_{dec}$ perfectly implements decapsulation of a ciphertext under pk, neither $\mathcal{O}_1$ nor $\mathcal{O}_2$ returns $\perp$ as a response to $\mathcal{A}_1$'s queries. Recall that the third condition of KEMs in the class $\mathcal{K}_{\mathsf{GS},n}$ guarantees that the function ψ_1 can be decomposed using the scheme-dependent functions $\{\psi_{1,i}\}_{i\in\{0,\dots,n\}}$ i.e. it is guaranteed that

$$\psi_1(pk, C, y_1, \dots, y_n) = \psi_{1,0}(pk, C) + \sum_{i\in[n]} \psi_{1,i}(pk, C) \cdot y_i$$

holds for any $pk = (\Lambda, aux, X_1, \dots, X_n) \in \mathcal{PK}$ and any ciphertext $C \in \mathbb{G} \times \{0,1\}^*$ (in the range of $\mathsf{Enc}(pk)$).

Next, $\mathcal{A}_1$ calculates $u'_j = u_j - \psi_{1,0}(pk, C_j)$ for every $j \in [n]$. Note that for each index $j \in [n]$, we must have

$$\sum_{i\in[n]} \psi_{1,i}(pk, C_j) \cdot y_i = u'_j. \tag{5}$$

For a ciphertext C, let $\boldsymbol{\psi}(pk, C) \in \mathbb{Z}_p^n$ be the n-dimensional (row) vector whose i-th element is $\psi_{1,i}(pk, C)$, i.e. $\boldsymbol{\psi}(pk, C) = (\psi_{1,1}(pk, C), \dots, \psi_{1,n}(pk, C))$. Let $\mathbf{C} = (C_1, \dots, C_n)$ be the ciphertexts returned from $\mathcal{O}_1$, and let $\boldsymbol{\Psi}(pk, \mathbf{C}) \in \mathbb{Z}_p^{n\times n}$ be the $n \times n$ matrix whose i-th row is $\boldsymbol{\psi}(pk, c_i)$. That is,

$$\boldsymbol{\Psi}(pk, \mathbf{C}) = \begin{pmatrix} \boldsymbol{\psi}(pk, C_1) \\ \boldsymbol{\psi}(pk, C_2) \\ \vdots \\ \boldsymbol{\psi}(pk, C_n) \end{pmatrix} = \begin{pmatrix} \psi_{1,1}(pk, C_1) & \psi_{1,2}(pk, C_1) & \dots & \psi_{1,n}(pk, C_1) \\ \psi_{1,1}(pk, C_2) & \psi_{1,2}(pk, C_2) & \dots & \psi_{1,n}(pk, C_2) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_{1,1}(pk, C_n) & \psi_{1,2}(pk, C_n) & \dots & \psi_{1,n}(pk, C_n) \end{pmatrix}$$

Furthermore, let $\mathbf{y}^{\mathrm{T}} = (y_1, \dots, y_n) \in \mathbb{Z}_p^n$ and $\mathbf{u}'^{\mathrm{T}} = (u'_1, \dots, u'_n) \in \mathbb{Z}_p^n$. Using this notation, we can describe the equation system that $\mathcal{A}_1$ has obtained as follows:

$$\boldsymbol{\Psi}(pk, \mathbf{C}) \cdot \mathbf{y} = \mathbf{u}',$$

where $\mathcal{A}_1$ knows all values in the matrix $\boldsymbol{\Psi}(pk, \mathbf{C})$ and the vector $\mathbf{u}'$. Note that this equation system is always consistent since the KEM Γ is assumed to be correct, pk is generated correctly in the experiment, and $(C_1, \dots, C_n)$ are valid ciphertexts under pk. Finally, $\mathcal{A}_1$ prepares the state information st consisting of all information known to $\mathcal{A}_1$ and terminates with output st.

$\mathcal{A}_2^{\mathcal{O}_1,\mathcal{O}_2}(\mathsf{st}, C^* = (c^*, d^*))$**:** $\mathcal{A}_2$ calculates the determinant $\det \boldsymbol{\Psi}(pk, \mathbf{C})$, and proceeds as follows:

- If $\det \mathbf{\Psi}(pk, \mathbf{C}) \neq 0$: $\mathcal{A}_2$ solves the equation system $\mathbf{\Psi}(pk, \mathbf{C}) \cdot \mathbf{y} = \mathbf{u}'$ and obtains $\mathbf{y}^{\mathrm{T}} = (y_1, \ldots, y_n)$ (which is possible because $\mathcal{A}_2$ knows all values in the matrix $\mathbf{\Psi}(pk, \mathbf{C})$ and the vector $\mathbf{u}'$). $\mathcal{A}_2$ then calculates the session-key $K^* = g^{\psi_0(pk, C^*, y_1, \ldots, y_n)} c^{*\psi_1(pk, C^*, y_1, \ldots, y_n)}$, and terminates with output K^*.
- If $\det \mathbf{\Psi}(pk, \mathbf{C}) = 0$: Using the algorithm guaranteed by Lemma 16, $\mathcal{A}_2$ checks if the vector $\boldsymbol{\psi}(pk, C^*)$ is linearly dependent on the $n-1$ vectors $\boldsymbol{\psi}(pk, C_1), \ldots, \boldsymbol{\psi}(pk, C_{n-1})$. If this is not the case, $\mathcal{A}_2$ gives up and aborts. Otherwise, the algorithm will recover constants $\alpha_1, \ldots, \alpha_{n-1} \in \mathbb{Z}_p$ such that

  $$\boldsymbol{\psi}(pk, C^*) = \alpha_1 \boldsymbol{\psi}(pk, C_1) + \ldots + \alpha_{n-1} \boldsymbol{\psi}(pk, C_{n-1})$$

  Here, notice that by multiplying (from the right) the column vector $\mathbf{y}$ to the above equation, and using the equation (5), we obtain:

  $$\sum_{i \in [n]} \psi_{1,i}(pk, C^*) \cdot y_i = \sum_{j \in [n-1]} \alpha_j u'_j$$

  Using the above, $\mathcal{A}_2$ computes the session-key as follows:

  $$\begin{aligned} K^* &= g^{\psi_{0,0}(pk, C^*)} \Big(\prod_{i \in [n]} X_i^{\psi_{0,i}(pk, C^*)} \Big) c^{*\psi_{1,0}(pk, C^*) + \sum_{j \in [n-1]} \alpha_j u'_j} \\ &= g^{\psi_{0,0}(pk, C^*) + \sum_{i \in [n]} \psi_{0,i}(pk, C^*) \cdot y_i} c^{*\psi_{1,0}(pk, C^*) + \sum_{i \in [n]} \psi_{1,i}(pk, C^*) \cdot y_i} \\ &= g^{\psi_0(pk, C^*, y_1, \ldots, y_n)} c^{*\psi_1(pk, C^*, y_1, \ldots, y_n)}, \end{aligned}$$

  where the last equation follows from the decomposition of the scheme-dependent functions ψ_0 and ψ_1 using $\{\psi_{i,j}\}_{i \in \{0,1\}, j \in [n]}$. Finally, $\mathcal{A}_2$ terminates with output K^*.

The above completes the description of $\mathcal{A}$. Note that $\mathcal{A}$ is deterministic. Furthermore, it is easy to see that $\mathcal{A}$ satisfies the property of an algebraic oracle algorithm: All group elements that $\mathcal{A}$ uses as inputs for the oracle queries either comes from the experiment, or from the oracles $\mathcal{O}$ and $\mathcal{O}_{dec}$ as answers to $\mathcal{A}$'s previous queries, and thus the corresponding extractor can be trivially constructed.

We now proceed to estimate $\mathcal{A}$'s success probability in the experiment $\widetilde{\mathsf{Expt}}_{\Gamma, \mathcal{A}^{\mathbb{D}}}^{\texttt{OW-}n\texttt{-CCA1}}(\lambda)$. Note that according to our description of $\mathcal{A}$, $\mathcal{A}$ succeeds in outputting the correct session-key K^* that corresponds to the challenge ciphertext C^* when either of the following two disjoint events occurs:

- (1) $\det \mathbf{\Psi}(pk, \mathbf{C}) \neq 0$
- (2) $\det \mathbf{\Psi}(pk, \mathbf{C}) = 0$ and $\exists \alpha_1, \ldots, \alpha_{n-1} \in \mathbb{Z}_p : \boldsymbol{\psi}(pk, C^*) = \sum_{j \in [n-1]} \alpha_i \boldsymbol{\psi}(pk, C_j)$

Let us denote by $p_1(pk)$ and $p_2(pk)$ the probabilities that the above events (1) and (2) occur, respectively, under the fixed public key $pk \in \mathcal{PK}$. That is, the probability space of $p_1(pk)$ and that of $p_2(pk)$ are over the choice of the oracle $\mathcal{O} \leftarrow \mathbb{D}$ and the calculation of the challenge ciphertext/session-key pair $(C^*, K^*) \leftarrow \mathsf{Enc}(pk)$. By definition, it holds that

$$\Pr[\widetilde{\mathsf{Expt}}_{\Gamma, \mathcal{A}^{\mathbb{D}}}^{\texttt{OW-}n\texttt{-CCA1}}(\lambda) = 1] = \mathop{\mathbf{E}}_{(pk, sk) \leftarrow \mathsf{KG}(\Lambda)} [p_1(pk) + p_2(pk)],$$

where KG is the key generation algorithm of the KEM Γ. Recall that by the definition of the oracle $\mathcal{O} = (\mathcal{O}_1, \mathcal{O}_2)$, if $\mathcal{O}$ is picked according to $\mathbb{D}$, then the function F used in $\mathcal{O}$ is a random function, and thus the values $(r_1, \ldots, r_n)$ output from $F(pk)$ are distributed uniformly over $\mathbb{Z}_p^n$. Furthermore, by definition the experiment chooses the randomness r^* used to generate the challenge ciphertext C^* uniformly from $\mathbb{Z}_p$. Therefore, if a public key pk is fixed in the experiment $\widetilde{\mathsf{Expt}}_{\Gamma, \mathcal{A}^{\mathbb{D}}}^{\texttt{OW-}n\texttt{-CCA1}}(\lambda)$, we can identify the remaining probability space of the experiment with the uniformly random choice of the randomness $(r_1, \ldots, r_n, r^*)$ from $\mathbb{Z}_p^{n+1}$ (the ciphertext/session-key pairs $\{(C_i, K_i)\}_{i \in [n]}$ and (C^*, K^*) are uniquely determined by pk and $(r_1, \ldots, r_n, r^*) \in \mathbb{Z}_p^{n+1}$).

In the following, we show a lower bound of $p_1(pk)+p_2(pk)$ under a fixed $pk \in \mathcal{PK}$, and thereby a lower bound of $\Pr[\widetilde{\mathsf{Expt}}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathbb{D}}}(\lambda) = 1]$.

The first probability $p_1(pk)$ (under a fixed $pk \in \mathcal{PK}$) is given by[5]:

$$p_1(pk) = \Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\det \mathbf{\Psi}(pk, \mathbf{C}) \neq 0]$$

Hence, it follows that

$$\Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\det \mathbf{\Psi}(pk, \mathbf{C}) = 0] = 1 - p_1(pk)$$

Before we estimate $p_2(pk)$, let us introduce the following: For a $pk \in \mathcal{PK}$, $i \in [n]$, an $(i-1)$-dimensional vector $(r_1, \dots, r_{i-1}) \in \mathbb{Z}_p^{i-1}$, and an element $r_i \in \mathbb{Z}_p$, let $\nu_i(pk, (r_1, \dots, r_{i-1}), r_i)$ be a variable indicating whether $\boldsymbol{\psi}(pk, C_i)$ is linearly dependent on the $n-1$ vectors $\boldsymbol{\psi}(pk, C_1), \dots, \boldsymbol{\psi}(pk, C_{i-1})$, i.e.

$$\nu_i(pk, (r_1, \dots, r_{i-1}), r_i) = \begin{cases} 1 \text{ if } \exists \alpha_1, \dots, \alpha_{i-1} \text{ s.t. } \boldsymbol{\psi}(pk, C_i) = \sum_{j \in [i-1]} \alpha_j \boldsymbol{\psi}(pk, r_j) \\ 0 \text{ otherwise} \end{cases}$$

For convenience, we let $\nu_1(pk, \emptyset, r_1)$ indicate whether or not $\boldsymbol{\psi}(pk, C_1) = \mathbf{0}$. To proceed, we will make use of the following claims.

Claim 20

$$\Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\det \mathbf{\Psi}(pk, \mathbf{C}) = 0] \leq \Pr_{r_1 \leftarrow \mathbb{Z}_p}[\nu_1(pk, \emptyset, r_1) = 1] + \ldots + \Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\nu_n(pk, (r_1, \dots, r_{n-1}), r_n) = 1].$$

Proof. (of Claim 20.) To see this, observe that, for a given choice of $(r_1, \dots, r_n) \leftarrow \mathbb{Z}_p^n$, $\det \mathbf{\Psi}(pk, \mathbf{C}) = 0$ implies that $\boldsymbol{\psi}(pk, C_1), \dots, \boldsymbol{\psi}(pk, C_n)$ are linearly dependent i.e. there must exist $\alpha_1, \dots, \alpha_n \in \mathbb{Z}_p$, not all zero, such that $\alpha_1 \boldsymbol{\psi}(pk, C_1) + \ldots + \alpha_n \boldsymbol{\psi}(pk, C_n) = \mathbf{0}$. Let i' be the largest index such that $\alpha_{i'} \neq 0$ and let $\alpha'_i = -\alpha_i / \alpha_{i'}$. Hence, we must have that

$$\boldsymbol{\psi}(pk, C_{i'}) = \alpha'_1 \boldsymbol{\psi}(pk, C_1) + \ldots + \alpha'_{i'-1} \boldsymbol{\psi}(pk, C_{i'-1})$$

and that $\nu_{i'}(pk, (r_1, \dots, r_{i'-1}), r_{i'}) = 1$. This implies that for any choice of $(r_1, \dots, r_n) \in \mathbb{Z}_p^n$, if $\det \mathbf{\Psi}(pk, \mathbf{C}) = 0$, there is at least one i for which $\nu_i(pk, (r_1, \dots, r_{i-1}), r_i) = 1$. Hence, it must hold that

$$\Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\det \mathbf{\Psi}(pk, \mathbf{C}) = 0] \leq \Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\nu_1(pk, \emptyset, r_1) = 1 \vee \cdots \vee \nu_n(pk, (r_1, \dots, r_{n-1}), r_n) = 1].$$

Applying the union bound to the right-hand side of the above equation yields the equation in Claim 20. $\square$

Claim 21 $\Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\nu_n(pk, (r_1, \dots, r_{n-1}), r_n) = 1] \geq (1 - p_1(pk))/n$.

Proof. (of Claim 21.) To see that this must be true, assume for the purpose of a contradiction that

$$\Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\nu_n(pk, (r_1, \dots, r_{n-1}), r_n) = 1] < (1 - p_1(pk))/n.$$

Note that

$$\begin{aligned}
&\Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\nu_n(pk, (r_1, \dots, r_{n-1}), r_n) = 1] \\
&= \Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\exists \alpha_1, \dots, \alpha_{n-1} : \boldsymbol{\psi}(pk, C_n) = \sum_{i \in [n-1]} \alpha_i \boldsymbol{\psi}(pk, C_i)] \\
&\geq \Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\exists \alpha_1, \dots, \alpha_{n-1} : \boldsymbol{\psi}(pk, C_n) = \sum_{i \in [n-1]} \alpha_i \boldsymbol{\psi}(pk, C_i) \wedge \alpha_{n-1} = 0] \\
&= \Pr_{r_1,\dots,r_{n-1} \leftarrow \mathbb{Z}_p}[\exists \alpha_1, \dots, \alpha_{n-2} : \boldsymbol{\psi}(pk, C_{n-1}) = \sum_{i \in [n-2]} \alpha_i \boldsymbol{\psi}(pk, C_i)] \\
&= \Pr_{r_1,\dots,r_{n-1} \leftarrow \mathbb{Z}_p}[\nu_{n-1}(pk, (r_1, \dots, r_{n-2}), r_{n-1}) = 1]
\end{aligned}$$

[5] Note that the event (1) is independent of the choice of the randomness r^* used for generating the challenge ciphertext/session-key pair.

Hence, assuming $\Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\nu_n(pk,(r_1,\dots,r_{n-1}),r_n)=1] < (1-p_1(pk))/n$, we must have

$$\Pr_{r_1,\dots,r_{n-1} \leftarrow \mathbb{Z}_p}[\nu_{n-1}(pk,(r_1,\dots,r_{n-2}),r_{n-1})=1] < (1-p_1(pk))/n.$$

Iterating the above argument yields that

$$\Pr_{r_1,\dots,r_i \leftarrow \mathbb{Z}_p}[\nu_i(pk,(r_1,\dots,r_{i-1}),r_i)=1] < (1-p_1(pk))/n$$

for any $i \in [n]$. Hence, from Claim 20 we get

$$\begin{aligned}\Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\det \mathbf{\Psi}(pk,\mathbf{C})=0] &\le \Pr_{r_1 \leftarrow \mathbb{Z}_p}[\nu_1(pk,\emptyset,r_1)=1] + \dots + \Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\nu_n(pk,(r_1,\dots,r_{n-1}),r_n)=1]\\ &< n \cdot (1-p_1(pk))/n\\ &= 1-p_1(pk)\end{aligned}$$

which contradicts that $\Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\det \mathbf{\Psi}(pk,\mathbf{C})=0] = 1-p_1(pk)$. This completes the proof of Claim 21. □

We will now return to the estimation of $p_2(pk)$ (under a fixed $pk \in \mathcal{PK}$):

$$\begin{aligned}p_2(pk) &= \Pr_{r_1,\dots,r_n,r^* \leftarrow \mathbb{Z}_p}[\det \mathbf{\Psi}(pk,\mathbf{C})=0 \wedge \nu_n(pk,(r_1,\dots,r_{n-1}),r^*)=1]\\ &= \mathop{\mathbf{E}}_{r_1,\dots,r_{n-1} \leftarrow \mathbb{Z}_p}\left[\Pr_{r_n,r^* \leftarrow \mathbb{Z}_p}[\det \mathbf{\Psi}(pk,\mathbf{C})=0 \wedge \nu_n(pk,(r_1,\dots,r_{n-1}),r^*)=1]\right]\\ &\overset{(*)}{=} \mathop{\mathbf{E}}_{r_1,\dots,r_{n-1} \leftarrow \mathbb{Z}_p}\left[\Pr_{r_n \leftarrow \mathbb{Z}_p}[\det \mathbf{\Psi}(pk,\mathbf{C})=0] \cdot \Pr_{r^* \leftarrow \mathbb{Z}_p}[\nu_n(pk,(r_1,\dots,r_{n-1}),r^*)=1]\right]\\ &\ge \mathop{\mathbf{E}}_{r_1,\dots,r_{n-1} \leftarrow \mathbb{Z}_p}\left[\left(\Pr_{r_n \leftarrow \mathbb{Z}_p}[\nu_n(pk,(r_1,\dots,r_{n-1}),r_n)=1]\right)^2\right]\\ &\ge \left(\mathop{\mathbf{E}}_{r_1,\dots,r_{n-1} \leftarrow \mathbb{Z}_p}\left[\Pr_{r_n \leftarrow \mathbb{Z}_p}[\nu_n(pk,(r_1,\dots,r_{n-1}),r_n)=1]\right]\right)^2\\ &= \left(\Pr_{r_1,\dots,r_n \leftarrow \mathbb{Z}_p}[\nu_n(pk,(r_1,\dots,r_{n-1}),r_n)=1]\right)^2\\ &\ge \left(\frac{1-p_1(pk)}{n}\right)^2\end{aligned}$$

where the equation (*) is because the considered events in this equation become independent once we fix pk and $(r_1,\dots,r_{n-1})$, the first inequality follows from the observation that $\nu_n(pk,(r_1,\dots,r_{n-1}),r_n)=1$ implies $\det \mathbf{\Psi}(pk,\mathbf{C})=0$ and therefore $\Pr_{r_n \leftarrow \mathbb{Z}_p}[\det \mathbf{\Psi}(pk,\mathbf{C})=0] \ge \Pr_{r_n \leftarrow \mathbb{Z}_p}[\nu_n(pk,(r_1,\dots,r_{n-1}),r_n)=1]$, the second inequality follows from Jensen's inequality[6], and the last inequality follows from Claim 21.

Now, under a fixed $pk \in \mathcal{PK}$, we have

$$p_1(pk)+p_2(pk) \ge p_1(pk) + \left(\frac{1-p_1(pk)}{n}\right)^2 \ge \frac{1}{n^2}\left(\left(p_1(pk)-\frac{1}{2}\right)^2+\frac{3}{4}\right) \ge \frac{3}{4n^2}.$$

Hence, we finally get

$$\mathop{\mathbf{E}}_{\mathcal{O} \leftarrow \mathbb{D}}[\mathsf{Adv}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathcal{O}}}(\lambda)] = \Pr[\widetilde{\mathsf{Expt}}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\Gamma,\mathcal{A}^{\mathbb{D}}}(\lambda)=1] = \mathop{\mathbf{E}}_{(pk,sk) \leftarrow \mathsf{KG}(\Lambda)}[p_1(pk)+p_2(pk)] \ge \frac{3}{4n^2},$$

which completes the proof of Lemma 9. □

[6] If X is a random variable and f is a convex function, then $\mathbf{E}[f(X)] \ge f(\mathbf{E}[X])$. We use $f(x)=x^2$.

D Proof of Lemma 10

Proof. Let GS, $\varGamma \in \mathcal{K}_{\mathsf{GS},n}$, $\mathsf{P} = (\mathsf{I}, \mathsf{U}, \mathsf{V})$, and $\mathbb{D}$ be as stated in the lemma. For an oracle algorithm $\mathcal{A}$, let us denote by $\widetilde{\mathsf{Expt}}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathbb{D}}}(\lambda)$ the experiment $\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathcal{O}}}(\lambda)$ where the choice of the oracle $\mathcal{O}$ according to the distribution $\mathbb{D}$ is also taken into account. That is,

$$\widetilde{\mathsf{Expt}}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathbb{D}}}(\lambda) : [\mathcal{O} \leftarrow \mathbb{D}; \text{Return } \mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathcal{O}}}(\lambda)]$$

Then, by definition,

$$\mathop{\mathbf{E}}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathcal{O}}}(\lambda)] = \mathop{\mathbf{E}}_{\mathcal{O}\leftarrow\mathbb{D}}[\Pr[\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathcal{O}}}(\lambda) = 1] - \delta(\lambda)] = \Pr[\widetilde{\mathsf{Expt}}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathbb{D}}}(\lambda) = 1] - \delta(\lambda).$$

Now, fix arbitrarily an algebraic oracle PPTA $\mathcal{A}$. Suppose $q = q(\lambda)$ is the number of oracle queries that $\mathcal{A}$ makes. (Since $\mathcal{A}$ is a PPTA, q is a polynomial, and without loss of generality, we assume $q > 0$.) Then, by definition of an algebraic oracle PPTA, there exists a corresponding "decomposition" algebraic PPTA $\widehat{\mathcal{A}}$ that satisfies the condition explained in Section 2.3, and $\widehat{\mathcal{A}}$ halts after q invocations of the oracle $\mathcal{O}$. Let $\widehat{\mathcal{E}}$ be the extractor corresponding to $\widehat{\mathcal{A}}$.

We consider the following sequence of games:

Game 1: This is the experiment $\widetilde{\mathsf{Expt}}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathbb{D}}}(\lambda)$ itself.

Game 2: Same as Game 1, except that in this game, any $\mathcal{O}_2$-query $(pk, (K_1, \dots, K_n), \sigma)$ that is not preceded by a $\mathcal{O}_1$-query pk (with the same pk) is answered with $\perp$.

Game 3: Same as Game 2, except that in this game, the experiment maintains a list L which is initially empty, and $\mathcal{A}$'s oracle queries are answered as follows:

- $\mathcal{O}_1$-query pk is answered using "lazy sampling" as follows:
  1. $\mathcal{O}_1$ returns $\perp$ if $pk \notin \mathcal{PK}$.
  2. If there is an entry of the form $(pk, \mathbf{C}, *, \sigma)$ for some $\mathbf{C} = (C_1, \dots, C_n)$ in the list L, $\mathcal{O}_1$ returns $(\mathbf{C}, \sigma)$ to $\mathcal{A}$.
  3. $\mathcal{O}_1$ picks $r_1, \dots r_n \in \mathbb{Z}_p$ and $\sigma \in \{0,1\}^\lambda$ uniformly at random, computes $(C_i, K_i) \leftarrow \mathsf{Enc}(pk; r_i)$ for every $i \in [n]$, and sets $\mathbf{C} = (C_1, \dots, C_n)$ and $\mathbf{K} = (K_1, \dots, K_n)$.
  4. $\mathcal{O}_1$ returns $(\mathbf{C}, \sigma)$ to $\mathcal{A}$, and stores the values $(pk, \mathbf{C}, \mathbf{K}, \sigma)$ in the list L.
- $\mathcal{O}_2$-query $(pk = (\varLambda, aux, (X_1, \dots, X_n)), \mathbf{K} = (K_1, \dots, K_n), \sigma)$ is answered using the list L and the extractor $\widehat{\mathcal{E}}$ for $\widehat{\mathcal{A}}$, where $\widehat{\mathcal{A}}$ is the "decomposition" of the algebraic oracle algorithm $\mathcal{A}$, in the following way:
  1. $\mathcal{O}_2$ checks if there is an entry of the form $(pk, *, \mathbf{K}, \sigma)$ in the list L, and returns $\perp$ if there is no such entry.
  2. Let $\mathbf{C} = (C_1, \dots, C_n)$ be the vector of ciphertexts retrieved from the entry found in the above step, st be the state information that $\widehat{\mathcal{A}}$ outputs with this $\mathcal{O}_2$-query, $\mathbf{X}'$ be the vector consisting of all group elements that are input to $\widehat{\mathcal{A}}$ until this point (either from the experiment or from $\mathcal{O}$ as answers to previous queries), and let $\mathbf{X}'_i = \mathbf{X}' \backslash \{c_i\}$, where c_i is the first component of the ciphertext $C_i = (c_i, d_i)$. For every $i \in [n]$, $\mathcal{O}_2$ parses C_i as $(c_i, d_i) \in \mathbb{G} \times \{0,1\}^n$ and runs the extractor $\widehat{\mathcal{E}}$ on $\widehat{\mathcal{A}}$'s input ($\varLambda$, $\mathbf{X}'$ and st) to obtain an integer $u_i \in \mathbb{Z}_p$ such that $K_i = c_i^{u_i} \cdot (\mathbf{X}'^{\mathbf{z}_i}_i)$ for some $\mathbf{z}_i \in \mathbb{Z}_p^{|\mathbf{X}'_i|}$.[7] [8]
  3. If for some $i \in [n]$ the extractor $\widehat{\mathcal{E}}$ fails to compute u_i or $u_i \neq \psi_1(pk, C_i, y_1, \dots, y_n)$, then $\mathcal{O}_2$ returns $\perp$ to $\mathcal{A}$.
  4. $\mathcal{O}_2$ sets $\mathbf{u} = (u_1, \dots, u_n)$ and returns $\mathbf{u}$ to $\mathcal{A}$.

Game 4: Same as Game 3, except that in the response to $\mathcal{O}_2$-queries from $\mathcal{A}$, $\mathcal{O}_2$ does not check if $u_i \neq \psi_1(pk, C_i, y_1, \dots, y_n)$ for any $i \in [t]$ in step 3, and directly uses the value u_i for $\mathbf{u}$.

[7] Recall that for $\mathbf{X} \in \mathbb{G}^n$ and $\mathbf{y} \in \mathbb{Z}_p^n$, $\mathbf{X}^{\mathbf{y}} = \prod_{i\in[|\mathbf{X}|]} \mathbf{X}[i]^{\mathbf{y}[i]}$.

[8] Note that according to the definition of $\mathcal{O}_1$ in Game 3, $\mathbf{C} = (C_1, \dots, C_n)$ must have appeared previously as an answer to one of $\mathcal{A}$'s $\mathcal{O}_1$-queries. Furthermore, recall that each ciphertext C_i is of the form $(c_i, d_i) \in \mathbb{G} \times \{0,1\}^*$. Therefore, $\mathbf{X}'$ must contain $(c_1, \dots, c_n)$.

For each $i \in [4]$, let Succ_i be the event that in Game i, $\mathcal{A}$ succeeds in solving the NIP P, namely, $\mathcal{A}^{\mathcal{O}}(1^\lambda, y)$ outputs x such that $\mathsf{V}(y, x, w) = \top$ in Game i.

Then, $\mathbf{E}_{\mathcal{O} \leftarrow \mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathcal{O}}}(\lambda)]$ can be estimated as follows.

$$\begin{aligned}
\mathop{\mathbf{E}}_{\mathcal{O} \leftarrow \mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathcal{O}}}(\lambda)] &= \Pr[\widetilde{\mathsf{Expt}}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathbb{D}}}(\lambda) = 1] - \delta(\lambda) \\
&= \Pr[\mathsf{Succ}_1] - \delta(\lambda) \\
&\leq \sum_{i \in [3]} |\Pr[\mathsf{Succ}_i] - \Pr[\mathsf{Succ}_{i+1}]| + \Pr[\mathsf{Succ}_4] - \delta(\lambda) \qquad (6)
\end{aligned}$$

In the following, we upperbound each term in the above inequality.

Claim 22 $|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]| \leq \frac{q}{2^\lambda - q}$.

Proof. (of Claim 22.) For $\gamma \in \{1, 2\}$, let Forge_γ be the event that in Game γ, $\mathcal{A}$ submits at least one $\mathcal{O}_2$-query $(pk, (K_1, \ldots, K_n), \sigma)$ such that the answer to this query in Game 1 is not $\perp$ and $\mathcal{A}$ has not previously asked an $\mathcal{O}_1$-query pk (with the same pk). Such query is answered with $\perp$ in Game 2, while it is answered with some value that is not $\perp$ in Game 1. Since Game 1 and Game 2 proceed identically unless Forge_1 or Forge_2 occurs, we have

$$|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]| \leq \Pr[\mathsf{Forge}_1] = \Pr[\mathsf{Forge}_2].$$

Thus, to prove the claim it is sufficient to upper-bound $\Pr[\mathsf{Forge}_1]$.

Note that if $\mathcal{O}$ is picked according to $\mathbb{D}$, then by definition, F used in the oracle $\mathcal{O}$ is a random function. Furthermore, to make a query that causes the event Forge_1 in Game 1, $\mathcal{A}$ has to come up with the tag $\sigma \in \{0,1\}^\lambda$ which is an output of the random function $F(pk)$ without ever making an $\mathcal{O}_1$-query pk. Consider the probability (in Game 1) that "$\mathcal{A}$'s i-th query is the first $\mathcal{O}_2$-query that causes the event Forge_1." Before the i-th query, $\mathcal{A}$ will at most have learned $i - 1$ strings[9] in $\{0,1\}^\lambda$ that are not the actual σ for a given pk. Therefore,

$$\Pr[i\text{-th query is the first } \mathcal{O}_2\text{-query that causes } \mathsf{Forge}_1] \leq \frac{1}{2^\lambda - (i-1)} \leq \frac{1}{2^\lambda - q}$$

where the latter inequality is because $i \leq q$. Then, by the union bound over the q queries by $\mathcal{A}$, we have $\Pr[\mathsf{Forge}_1] \leq \frac{q}{2^\lambda - q}$. This completes the proof of Claim 22. □

Claim 23 $|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_3]|$ *is negligible.*

Proof. (of Claim 23.) Note that the difference between Game 2 and Game 3 is only in the oracles which $\mathcal{A}$ is given access to. It is easy to see that the responses to $\mathcal{O}_1$-queries are distributed identically in both Game 2 and Game 3, because the lazy sampling can simulate the random function F used by $\mathcal{O}_1$ (and $\mathcal{O}_2$) perfectly for PPTAs. (Recall that in Game 2 and Game 3, any $\mathcal{O}_2$-query using pk that is not preceded by $\mathcal{O}_1$-query with the same pk is always answered with $\perp$.) As for $\mathcal{O}_2$, if $\mathcal{O}_2$ on input $(pk, \mathbf{K}, \sigma)$ outputs some value that is not $\perp$ in Game 3, then it must agree with the output in Game 2 due to the check $u_i = \psi_1(pk, C_i, y_1, \ldots, y_n)$ performed in the step 3. In particular, although $\mathcal{O}_2$ in Game 3 does not re-compute the ciphertext/session-key pairs $\{(c_i, K_i)\}_{i \in [n]}$, checking whether $(pk, *, \mathbf{K}, \sigma) \in L$ is equivalent to checking whether the session-keys $\mathbf{K} = (K_1, \ldots, K_n)$ in this $\mathcal{O}_2$-query and those generated during the response to the previously asked $\mathcal{O}_1$-query pk agree. The only part where the values output from $\mathcal{O}_2$ in Game 2 and in Game 3 can disagree is in step 3 of $\mathcal{O}_2$ in Game 3 where the extractor $\widehat{\mathcal{E}}$ could fail to compute u_i for some $i \in [n]$. Let Fail_3 be the event that (in Game 3), $\widehat{\mathcal{E}}$ fails to compute u_i for some $i \in [n]$ for some $\mathcal{O}_2$-query. Then, we have

$$|\Pr[\mathsf{Succ}_2] - \Pr[\mathsf{Succ}_3]| \leq \Pr[\mathsf{Fail}_3].$$

[9] These strings might be learned by making $\mathcal{O}_2$-queries with a same pk (without using it as a $\mathcal{O}_1$-query).

However, by the definition of an algebraic oracle algorithm, for each invocation, the probability that the corresponding extractor fails is negligible. $\mathcal{A}$ can make only polynomially many $\mathcal{O}_2$-queries, each of which can contain at most n session-keys, and thus the union bound over the possible session-keys contained in $\mathcal{A}$'s $\mathcal{O}_2$-queries in Game 3 (which is at most polynomial) shows that $\Pr[\mathsf{Fail}_3]$ is negligible. This completes the proof of Claim 23. $\square$

Claim 24 *There exists a PPTA $\mathcal{B}_1$ and a negligible function $\mu'(\lambda)$ such that*

$$|\Pr[\mathsf{Succ}_3] - \Pr[\mathsf{Succ}_4]| \leq nq \cdot \mathsf{Adv}^{\mathtt{DL}}_{\mathsf{GS},\mathcal{B}_1}(\lambda) + \mu'(\lambda).$$

Proof. (of Claim 24.) Note that the difference between Game 3 and Game 4 is whether $u_i = \psi_1(pk, C_i, y_1, \ldots, y_n)$ is checked in the step 3 of $\mathcal{O}_2$. For $\gamma \in \{3, 4\}$, let Bad_γ be the event that in Game γ, $\mathcal{A}$ submits at least one $\mathcal{O}_2$-query $(pk = (\varLambda, aux, X_1, \ldots, X_n), \mathbf{K}, \sigma)$ such that it passes the checks in the step 1, and $u_i \neq \psi_i(pk, C_i, y_1, \ldots, y_n)$ for some $i \in [n]$, where u_i is computed by the extractor $\widehat{\mathcal{E}}$ and $y_i = \log_g X_i$ for $i \in [n]$. Note that if a query passes the check in the step 1 of $\mathcal{O}_2$ in Game 3 (resp. Game 4), it means that $\mathcal{A}$ must have submitted the $\mathcal{O}_1$-query pk before the $\mathcal{O}_2$-query that caused the event Bad_3 (resp. Bad_4).

Since Game 3 and Game 4 proceed identically unless Bad_3 or Bad_4 occurs, we have

$$|\Pr[\mathsf{Succ}_3] - \Pr[\mathsf{Succ}_4]| \leq \Pr[\mathsf{Bad}_3] = \Pr[\mathsf{Bad}_4].$$

We show that there exists a PPTA $\mathcal{B}_1$ such that $\mathsf{Adv}^{\mathtt{DL}}_{\mathsf{GS},\mathcal{B}_1}(\lambda) \geq \frac{1}{nq} \cdot \Pr[\mathsf{Bad}_4] - \mu''(\lambda)$ for some negligible function $\mu''(\lambda)$, which proves the claim by setting $\mu'(\lambda) = nq \cdot \mu''(\lambda)$. $\mathcal{B}_1$ uses $\mathcal{A}$ (actually, $\widehat{\mathcal{A}}$) as a subroutine and runs as follows:

$\mathcal{B}_1(1^\lambda, \varLambda = (g, p, \mathbb{G}), Z = g^\alpha)$: (where $\alpha \in \mathbb{Z}_p$ is chosen uniformly at random and unknown to $\mathcal{B}_1$) $\mathcal{B}_1$ picks two indices $i^* \in [q]$ and $j^* \in [n]$ uniformly at random, runs $(y, w) \leftarrow \mathsf{I}(\varLambda)$, and generates an empty list L (which will be used to simulate $\mathcal{O}_1$ and $\mathcal{O}_2$ as in Game 4). Then $\mathcal{B}_1$ picks a randomness $r_\mathcal{A}$ (for $\mathcal{A}$) and then runs $\mathcal{A}(1^\lambda, y; r_\mathcal{A})$.
For $1 \leq i < i^*$, $\mathcal{B}_1$ answers $\mathcal{A}$'s i-th query exactly as in Game 4, which is possible because $\mathcal{O}_1$-queries can be answered perfectly by lazy sampling (which $\mathcal{B}_1$ implements using the list L), and $\mathcal{O}_2$-queries can be answered by using L and the extractor $\widehat{\mathcal{E}}$ (recall that $\mathcal{B}_1$ holds the randomness $r_\mathcal{A}$ which is used to run $\mathcal{A}$).
When $\mathcal{A}$ makes the i^*-th query, $\mathcal{B}_1$ proceeds as follows:

1. $\mathcal{B}_1$ gives up and aborts if either of the following occurs: (1) this is not an $\mathcal{O}_1$-query, (2) This is an $\mathcal{O}_1$-query pk but $pk \notin \mathcal{PK}$, or there already exists an entry of the form $(pk, *, *, *)$ in L.
2. Let $pk^* = (\varLambda, aux^*, X_1^*, \ldots, X_n^*)$ be the public key in this $\mathcal{O}_1$-query. $\mathcal{B}_1$ recovers the discrete logarithms $\{y_i^* = \log_g X_i^*\}_{i \in [n]}$ by using the extractor $\widehat{\mathcal{E}}$. Note that this is possible (unless the extractor fails) since $\mathcal{B}_1$ can compute the discrete logarithms (with respect to g) of all group elements that are input to $\mathcal{A}$ until this point.[10] $\mathcal{B}_1$ gives up and aborts if the extractor fails during the above computation of the discrete logarithms.
3. $\mathcal{B}_1$ picks $(r_1^*, \ldots, r_{j^*-1}^*, r_{j^*+1}^*, \ldots, r_n^*) \in \mathbb{Z}_p^{n-1}$ and $\sigma^* \in \{0,1\}^\lambda$ uniformly at random, and calculates $(C_i^*, K_i^*) \leftarrow \mathsf{Enc}(pk^*; r_i^*)$ for all $i \in [n] \backslash \{j^*\}$
4. $\mathcal{B}_1$ regards Z as $c_{j^*}^*$, and calculates $d_{j^*}^* \leftarrow \widetilde{\psi}(pk^*, c_{j^*}^*, y_1^*, \ldots, y_n^*)$, where $\widetilde{\psi}$ is the scheme-dependent function of $\varGamma$ that is guaranteed by the fourth property of the class $\mathcal{K}_{\mathsf{GS},n}$. $\mathcal{B}_1$ also sets $C_{j^*}^* \leftarrow (c_{j^*}^*, d_{j^*}^*)$. Here, note that $C_{j^*}^*$ is a correct ciphertext under pk^* using the randomness α, i.e. it holds that $C_{j^*}^* = \mathsf{Enc}(pk^*; \alpha)$ where $\alpha = \log_g {c^*}_{j^*} = \log_g Z$.

[10] It should be noticed that the instance y given to $\mathcal{A}$ could contain group elements. However, since I is an algebraic PPTA, $\mathcal{B}_1$ can obtain all discrete logarithms (with respect to g) of the group elements contained in the instance y by running the extractor corresponding to I. Hence, at this point, the only group element whose discrete logarithm (with respect to g) is unknown to $\mathcal{B}_1$ is Z, but this group element has not been given to $\mathcal{A}$ up until this point.

5. $\mathcal{B}_1$ calculates $K^*_{j^*} = g^{\psi_0(pk^*, C^*_{j^*}, y^*_1, \ldots, y^*_n)} \cdot c^*_{j^*}{}^{\psi_1(pk^*, C^*_{j^*}, y^*_1, \ldots, y^*_n)}$. Note that by the third property of the class $\mathcal{K}_{\mathsf{GS},n}$, $K^*_{j^*}$ computed here is a correct session-key corresponding to $C^*_{j^*}$.
6. $\mathcal{B}_1$ sets $\mathbf{C}^* \leftarrow (C^*_1, \ldots, C^*_n)$ and $\mathbf{K}^* \leftarrow (K^*_1, \ldots, K^*_n)$, returns $(\mathbf{C}^*, \sigma^*)$ to $\mathcal{A}$, and stores the values $(pk^*, \mathbf{C}^*, \mathbf{K}^*, \sigma^*)$ in the list L.

After the i^*-th query, $\mathcal{B}_1$ continues simulating the answers to oracle queries from $\mathcal{A}$ as in Game 4 until $\mathcal{A}$ terminates.

At some point, $\mathcal{A}$ terminates (with some output which $\mathcal{B}_1$ discards). $\mathcal{B}_1$ checks if $\mathcal{A}$ has issued an $\mathcal{O}_2$-query of the form $(pk^*, \mathbf{K}^*, \sigma^*)$ that (1) caused the event Bad_4[11] and (2) for which $u_{j^*} \neq \psi_1(pk^*, C^*_{j^*}, y^*_1, \ldots, y^*_n)$[12], where u_{j^*} is obtained by $\mathcal{B}_1$ by running the extractor $\widehat{\mathcal{E}}$ as specified in Game 4. If no such query has been made by $\mathcal{A}$, $\mathcal{B}_1$ simply gives up and aborts. If such query is found, $\mathcal{B}_1$ calculates the value $s \in \mathbb{Z}_p$ such that $K^*_{j^*} = g^s Z^{u_{j^*}}$. Note that $\mathcal{B}_1$ will be able to compute this s efficiently since $\mathcal{B}_1$ has obtained all discrete logarithms of the group elements that are input to $\mathcal{A}$, except for Z, through $\widehat{\mathcal{E}}$ (unless $\widehat{\mathcal{E}}$ failed in the response to one of the previous queries). Since the first component of the j^*-th ciphertext $C^*_{j^*}$ in $\mathbf{C}^*$ is Z, we must have that $K^*_{j^*} = g^s Z^{u_{j^*}} = g^{\psi_0(pk^*, C^*_{j^*}, y^*_1, \ldots, y^*_n)} Z^{\psi_1(pk^*, C^*_{j^*}, y^*_1, \ldots, y^*_n)}$ and $u_{j^*} \neq \psi_1(pk^*, C^*_{j^*}, y^*_1, \ldots, y^*_n)$. Then, $\mathcal{B}_1$ calculates $\alpha = \log_g Z = (\psi_0(pk^*, C^*_{j^*}, y^*_1, \ldots, y^*_n) - s)/(u_{j^*} - \psi_1(pk^*, C^*_{j^*}, y^*_1, \ldots, y^*_n))$, and terminates with output α.

It is straightforward to see that $\mathcal{B}_1$ simulates Game 4 perfectly for $\mathcal{A}$ as long as $\mathcal{A}$'s i^*-th query is a fresh $\mathcal{O}_1$-query with a valid public-key $pk^* \in \mathcal{PK}$ and the extractor $\widehat{\mathcal{E}}$ that $\mathcal{B}_1$ runs during the response to the i^*-th query does not fail. In particular, since the j^*-th ciphertext $C^*_{j^*}$ of $\mathbf{C}^*$ that is used for answering $\mathcal{A}$'s i^*-th query is generated from the DL instance Z, $C^*_{j^*}$ is a correct ciphertext under pk^* using the randomness $\alpha = \log_g, Z$ which, by the definition of the DL problem, is distributed uniformly in $\mathbb{Z}_p$. Therefore, $C^*_{j^*}$ is distributed identically to a ciphertext generated from $\mathsf{Enc}(pk^*)$, and thus $\mathcal{B}_1$'s simulated response to the i^*-th query is perfect (assuming this query is a fresh and valid $\mathcal{O}_1$-query $pk^* \in \mathcal{PK}$ and the extractor does not fail). Moreover, once $\mathcal{A}$ makes an $\mathcal{O}_2$-query that causes the event Bad_4, $\mathcal{B}_1$ will be able to compute the discrete logarithm $\alpha = \log_g Z$ with probability at least $\frac{1}{nq}$ (except some negligible failure probability due to the extractor $\widehat{\mathcal{E}}$), because $\mathcal{B}_1$'s guess on which query and which index the event Bad_4 occurs is correct with probability at least $\frac{1}{nq}$, and once $\mathcal{B}_1$ succeeds in guessing the position, it succeeds in outputting the discrete logarithm of Z with overwhelming probability (the error can occur when the extractor $\widehat{\mathcal{E}}$ fails). Therefore, there exists a negligible function $\mu''(\lambda)$, indicating the total failure probability of the extractor $\widehat{\mathcal{E}}$, such that

$$\mathsf{Adv}^{\mathsf{DL}}_{\mathsf{GS},\mathcal{B}_1}(\lambda) \geq \frac{1}{nq} \cdot \Pr[\mathsf{Bad}_4] - \mu''(\lambda),$$

which completes the proof of Claim 24. □

Claim 25 *There exists a PPTA $\mathcal{B}_2$ such that* $\Pr[\mathsf{Succ}_4] - \delta(\lambda) = \mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{B}_2}(\lambda)$.

Proof. (of Claim 25.) We show how to construct a PPTA $\mathcal{B}_2$ that perfectly simulates Game 4 for $\mathcal{A}$ and solves the NIP P with probability exactly $\Pr[\mathsf{Succ}_4]$. The description of $\mathcal{B}_2$ is as follows:

$\mathcal{B}_2(1^\lambda, y)$**:** $\mathcal{B}_2$ generates an empty list L (which is used to simulate $\mathcal{O}_1$ and $\mathcal{O}_2$ as in Game 4), picks a randomness $r_\mathcal{A}$ that is used for $\mathcal{A}$, and runs $\mathcal{A}(1^\lambda, y; r_\mathcal{A})$. $\mathcal{B}_2$ answers to all queries from $\mathcal{A}$ as in Game 4, which is possible because $\mathcal{O}_1$-queries can be perfectly answered by the lazy sampling using the list L, and $\mathcal{O}_2$-queries can be answered by using L and the extractor $\widehat{\mathcal{E}}$ (recall that $\mathcal{B}_2$ holds the randomness $r_\mathcal{A}$ used to run $\mathcal{A}$). When $\mathcal{A}$ outputs x, $\mathcal{B}_2$ also outputs x and terminates.

[11] Note that unless $\mathcal{B}_1$ aborts due to the extractor $\widehat{\mathcal{E}}$ failing, $\mathcal{B}_1$ will know all discrete logarithms $\{y^*_i = \log_g X^*_i\}_{i \in [n]}$, and will hence be able to check whether there is an $\mathcal{O}_2$-query that causes the event Bad_4 efficiently by computing the scheme-dependent function ψ_1.

[12] Note that even if it is a query that causes Bad_4, the position j such that $u_j \neq \psi_1(pk^*, C^*_j, y^*_1, \ldots, y^*_n)$ could be different from j^*.

It is easy to see that $\mathcal{B}_2$ perfectly simulates Game 4 for $\mathcal{A}$, and thus $\mathcal{B}_2$ succeeds in outputting x such that $\mathsf{V}(y, x, w) = \top$ whenever $\mathcal{A}$ does so (in Game 4), and thus we have $\Pr[\mathsf{Succ}_4] = \Pr[\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\mathcal{B}_2}(\lambda) = 1]$. This implies

$$\Pr[\mathsf{Succ}_4] - \delta(\lambda) = \Pr[\mathsf{Expt}^{\mathsf{P}}_{\mathsf{GS},\mathcal{B}_2}(\lambda) = 1] - \delta(\lambda) = \mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{B}_2}(\lambda),$$

which completes the proof of Claim 25. □

Using Claims 22 to 25 in the inequality (6), we conclude that there exist PPTAs $\mathcal{B}_1$ and $\mathcal{B}_2$, a polynomial $Q(\lambda)$, and a negligible function $\mu(\lambda)$ such that

$$\mathop{\mathbf{E}}_{\mathcal{O} \leftarrow \mathbb{D}}[\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}^{\mathcal{O}}}(\lambda)] \leq Q(\lambda) \cdot \mathsf{Adv}^{\mathtt{DL}}_{\mathsf{GS},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{B}_2}(\lambda) + \mu(\lambda).$$

Finally note that the proof works for any group scheme GS, any KEM $\Gamma \in \mathcal{K}_{\mathsf{GS},n}$, any NIP P with respect to GS, and any algebraic oracle PPTA $\mathcal{A}$. This completes the proof of Lemma 10. □

E Proof of Theorem 12

Proof. Let $\mathcal{A}$ be an adversary against the $\mathtt{IND}\text{-}q\text{-}\mathtt{CCA2}$ security of the proposed KEM Γ. Now consider the following sequence of games:

Game 1 This is the original $\mathtt{IND}\text{-}q\text{-}\mathtt{CCA2}$ security experiment $\mathsf{Expt}^{\mathtt{IND}\text{-}q\text{-}\mathtt{CCA2}}_{\Gamma,\mathcal{A}}(\lambda)$.

Game 2 In this game, we change the computation of the public key pk of Γ and the decapsulation oracle as follows:

- The public key pk is generated by firstly picking $x, y \leftarrow \mathbb{Z}_p$ uniformly at random, computing $(\kappa, \tau) \leftarrow \mathsf{HTrapGen}(\Lambda, g^x, g^y)$, and setting $pk \leftarrow (\Lambda, \kappa)$.
- The decapsulation oracle will, on input a ciphertext c, compute $(a_c, b_c) \leftarrow \mathsf{HTrapEval}(\tau, c)$ and return the key $K = c^{xa_c + yb_c}$.

The remaining part is unchanged from Game 1.

Let Succ_i be the event that $\mathcal{A}$ succeeds in correctly guessing the challenge bit in Game i. Then the advantage of $\mathcal{A}$ can be expressed as

$$\mathsf{Adv}^{\mathtt{IND}\text{-}q\text{-}\mathtt{CCA2}}_{\Gamma,\mathcal{A}}(\lambda) = \Pr[\mathsf{Succ}_1] - \frac{1}{2} \leq |\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]| + \Pr[\mathsf{Succ}_2] - \frac{1}{2}$$

In the following claims, we will bound each term in the right hand side of the above inequality.

Claim 26 $|\Pr[\mathsf{Succ}_1] - \Pr[\mathsf{Succ}_2]|$ *is negligible.*

Proof. (of Claim 26) Note that the distribution of the hash key κ returned by $\mathsf{HTrapGen}$ is statistically close to the κ returned by HGen. Hence, the distribution of the public key pk and the challenge ciphertext/session-key pair (c^*, K_b^*) in Game 1 is statistically close to that of Game 2. Furthermore, note that the output of the decapsulation oracle agree in Game 1 and Game 2 on all ciphertexts c. Specifically, let $c = g^r$ for some $r \in \mathbb{Z}_p$ and let $(a_c, b_c) \leftarrow \mathsf{HTrapEval}(\tau, c)$. In Game 2, the session-key K returned by the decapsulation oracle satisfies

$$K = c^{xa_c + yb_c} = (g^r)^{xa_c + yb_c} = \left((g^x)^{a_c}(g^y)^{b_c}\right)^r = \mathsf{Eval}(\kappa, c)^r$$

where the last equality is due to the property of $\mathsf{HTrapEval}$. This completes the proof of Claim 26. □

Claim 27 *There exist a PPTA $\mathcal{B}$ and a noticeable function $\mu(\cdot)$ with the following properties: (1) $\mathcal{B}$ is an algebraic oracle algorithm with respect to GS that treats $\mathcal{A}$ as an oracle, and (2) it holds that* $\mathsf{Adv}^{\mathtt{DDH}}_{\mathsf{GS},\mathcal{B}}(\lambda) \geq \mu(\lambda) \cdot (\Pr[\mathsf{Succ}_2] - 1/2)$.

Proof. (of Claim 27) We will construct a PPT algorithm $\mathcal{B}$ which, using $\mathcal{A}$ as a building block, solves the DDH problem with respect to GS with the claimed advantage. Initially, $\mathcal{B}$ receives a group description $\Lambda = (g, p, \mathbb{G})$ and an instance (g^α, g^β, Z) of the DDH problem, where Z is either a random element in $\mathbb{G}$ or $Z = g^{\alpha\beta}$. The goal of $\mathcal{B}$ is to output the bit 0 if Z is random in $\mathbb{G}$, and 1 if $Z = g^{\alpha\beta}$. $\mathcal{B}$ is constructed as follows.

$\mathcal{B}(\Lambda = (g, p, \mathbb{G}), g^\alpha, g^\beta, Z)$**:** $\mathcal{B}$ regards α as x in Game 2 (although $\mathcal{B}$ does not know the exact value of α). $\mathcal{B}$ picks $y \leftarrow \mathbb{Z}_p$ uniformly at random, runs $(\kappa, \tau) \leftarrow \mathsf{HTrapGen}(g^\alpha, g^y)$, sets $pk \leftarrow (\Lambda, \kappa)$, and runs $\mathcal{A}_1(pk)$.

When $\mathcal{A}_1$ submits a decapsulation query c, $\mathcal{B}$ proceeds as follows: $\mathcal{B}$ computes $(a_c, b_c) \leftarrow \mathsf{HTrapEval}(\tau, c)$. If $a_c \neq 0$, $\mathcal{B}$ aborts and outputs a random bit. Otherwise (i.e. $a_c = 0$), $\mathcal{B}$ computes the key $K = c^{yb_c}$. Note that in this case, the key K is computed correctly as in Game 2 since letting $c = g^r$ for some $r \in \mathbb{Z}_p$, we have $K = c^{yb_c} = ((g^\alpha)^{a_c}(g^y)^{b_c})^r = \mathsf{Eval}(\kappa, c)^r$.

At some point, $\mathcal{A}_1$ will terminate with state information st. $\mathcal{B}$ then sets the challenge encapsulation to be $c^* \leftarrow g^\beta$ and computes $(a_{c^*}, b_{c^*}) \leftarrow \mathsf{HTrapEval}(\tau, c^*)$. If $a_{c^*} = 0$, $\mathcal{B}$ aborts and outputs a random bit. Otherwise (i.e. $a_{c^*} \neq 0$), $\mathcal{B}$ computes the challenge session-key as $K^* \leftarrow Z^{a_{c^*}}(g^\beta)^{yb_{c^*}}$. Note that if $Z = g^{\alpha\beta}$, we must have

$$K^* = (g^{\alpha\beta})^{a_{c^*}}(g^\beta)^{yb_{c^*}} = ((g^\alpha)^{a_{c^*}}(g^y)^{b_{c^*}})^\beta = \mathsf{Eval}(\kappa, c^*)^\beta$$

which corresponds to the session-key encapsulated by $c^* = g^\beta$ in Game 2. On the other hand, if Z is random in $\mathbb{G}$, then so is K^*. $\mathcal{B}$ then runs $\mathcal{A}_2$ with input (st, c^*, K^*).

If $\mathcal{A}_2$ submits a decapsulation query, $\mathcal{B}$ will respond as in a decapsulation query submitted by $\mathcal{A}_1$. Note that $\mathcal{A}_1$ and $\mathcal{A}_2$ are allowed to ask at most q decapsulation queries in total.

Eventually, $\mathcal{A}_2$ will terminate with output a bit which $\mathcal{B}$ simply outputs as his answer to the problem instance (g^α, g^β, Z).

The above completes the description of $\mathcal{B}$. It is easily seen that $\mathcal{B}$ is an algebraic oracle algorithm with respect to GS which treats the adversary $\mathcal{A}$ as an oracle.

Let Abort be the event that $\mathcal{B}$ aborts in the experiment, and let $\mathsf{Succ}_\mathcal{B}$ be the event that $\mathcal{B}$ succeeds in solving the DDH problem. The advantage of $\mathcal{B}$ in solving the DDH problem is given by

$$\begin{aligned}\mathsf{Adv}^{\mathtt{DDH}}_{\mathsf{GS},\mathcal{B}}(\lambda) &= \Pr[\mathsf{Succ}_\mathcal{B}] - \frac{1}{2} \\ &= \Pr[\mathsf{Succ}_\mathcal{B}|\overline{\mathsf{Abort}}] \cdot \Pr[\overline{\mathsf{Abort}}] + \Pr[\mathsf{Succ}_\mathcal{B}|\mathsf{Abort}] \cdot \Pr[\mathsf{Abort}] - \frac{1}{2}\end{aligned}$$

Since $\mathcal{B}$ provides a perfect simulation of Game 2 for $\mathcal{A}$ when Abort does not occur, we must have $\Pr[\mathsf{Succ}_\mathcal{B}|\overline{\mathsf{Abort}}] = \Pr[\mathsf{Succ}_2]$. Furthermore, $\mathcal{B}$ will output a random bit when Abort occurs and therefore, $\Pr[\mathsf{Succ}_\mathcal{B}|\mathsf{Abort}] = 1/2$. Hence,

$$\begin{aligned}\mathsf{Adv}^{\mathtt{DDH}}_{\mathsf{GS},\mathcal{B}}(\lambda) &= \Pr[\mathsf{Succ}_2] \cdot \Pr[\overline{\mathsf{Abort}}] + \frac{1}{2}\Pr[\mathsf{Abort}] - \frac{1}{2} \\ &= \Pr[\overline{\mathsf{Abort}}] \cdot \left(\Pr[\mathsf{Succ}_2] - \frac{1}{2}\right)\end{aligned}$$

According to the conditions in which $\mathcal{B}$ aborts, we have

$$\Pr[\overline{\mathsf{Abort}}] = \Pr[a_{c_1} = \cdots = a_{c_q} = 0 \wedge a_{c^*} \neq 0],$$

where c_i denotes the i-th decapsulation query, $(a_{c_i}, b_{c_i}) \leftarrow \mathsf{HTrapEval}(\tau, c_i)$, and $(a_{c^*}, b_{c^*}) \leftarrow \mathsf{HTrapEval}(\tau, c^*)$. It follows directly from the assumption that H is a $(q, 1)$-programmable hash function that this probability is noticeable. Therefore, Claim 27 follows by letting $\mu(\lambda)$ denote the probability $\Pr[\overline{\mathsf{Abort}}]$. □

Combining the above two claims yields that the KEM Γ can be proven to be `IND-`q`-CCA2` secure under the DDH assumption via an algebraic black-box reduction. This completes the proof of Theorem 12. □