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 discuss the (im)possibility of constructing chosen ciphertext secure (CCA secure) 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 used to construct a simple, efficient and CCA secure KEM based on the hardness of the decisional Diffie-Hellman problem with a 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.

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 [10]. In this approach, a public key component, referred to as a key encapsulation mechanism (KEM), is used to encrypt a random session key, and the 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. Specifically, if the KEM achieves CCA security, a redundancy-free DEM can be used [27], since only one-time CCA security is required of the DEM to obtain a CCA secure hybrid encryption scheme. If the KEM achieves the slightly weaker notion of constrained CCA security [20], an authenticated DEM is required, which will introduce a small additional overhead corresponding to a message authentication code (MAC).[1]

The currently most efficient (constrained) CCA secure KEMs, which are provably secure in the standard model, are defined in prime order groups, and have a ciphertext overhead of at least two group elements, e.g. [10, 25, 4, 24, 20, 15]. An overview of these is given in Table 1. Note that when considering constrained CCA secure KEMs, the additional ciphertext overhead of a MAC must be taken into account.[2] The KEM by Cramer et al. [9], which achieves a ciphertext overhead of a single group element, can only be shown q-bounded CCA secure (for a predetermined number of decryption queries q), and hence will not lead to a fully CCA secure encryption scheme if combined with a DEM.

More Efficient Schemes? Given the existing KEMs, it is natural to ask: *Is it possible to construct a CCA secure KEM with a ciphertext overhead of less than two group elements?* Note that, besides being defined in prime order groups, the KEMs in Table 1 share some structural properties. More specifically, all of the KEMs include a random group element as part of the ciphertext which will be used to derive the session-key in the decapsulation. The remaining element(s) (except in KD [25], but see footnote[3]) are used to decide whether the ciphertext is accepted as "valid", but does not otherwise contribute to the computation of the decapsulated key.

Given this, 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 the above mentioned schemes. To illustrate this approach, consider a KEM by Cramer and Shoup [10, Sect. 9.3]. In this KEM, 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 consists of $(c_1 = g^r, c_2 = X_1^r, c_3 = X_2^r X_3^{r\alpha})$ and the corresponding

[1] Alternatively, the KEM can be generically converted to a CCA secure KEM [2], but this will likewise introduce an additional overhead of a MAC.

[2] 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, a MAC contributes with only λ bits.

[3] While the KEM in [25] makes use of "implicit rejection" and does not explicitly check the validity of a 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.

Table 1. The currently most efficient KEMs in terms of ciphertext overhead. The scheme Kiltz [24]† is identical to Kiltz [24], except that no hash function is used to derive the session-key (see Sect. 4.2 of [24]). In the table, CCCA denotes constrained CCA [20] and q-CCA denotes q-bounded CCA [9]. 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 [10]	IND-CCA	DDH	$3\|\mathbb{G}\|$
HaKu [15, Sect. 5]	IND-CCA	CDH	$3\|\mathbb{G}\|$
HaKu [15, Sect. 4.1]	OW-CCA	CDH	$3\|\mathbb{G}\|$
KD [25]	IND-CCCA	DDH	$2\|\mathbb{G}\|$
HoKi [20]	IND-CCCA	DDH	$2\|\mathbb{G}\|$
HaKu [15, Sect. 6]	IND-CCCA	DDH	$2\|\mathbb{G}\|$
Kiltz [24]	IND-CCA	GHDH	$2\|\mathbb{G}\|$
Kiltz [24]†	OW-CCA	GDH	$2\|\mathbb{G}\|$
BMW [4]	IND-CCA	DBDH	$2\|\mathbb{G}\|$
CHH+ [9]	IND-q-CCA	DDH	$1\|\mathbb{G}\|$

session-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 checks if $c_1^{x_1+y_1\alpha} c_2^{x_2+y_2\alpha} = c_3$, and outputs the session-key $K = c_1^z$ if this is the case. 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 [11]. This reduces the ciphertext overhead to match that of the other KEMs 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 [25], Hofheinz-Kiltz [20] and Hanaoka-Kurosawa [15].

A similar approach can be used to reduce the ciphertext overhead of the schemes Hofheinz-Kiltz [20], Hanaoka-Kurosawa [15], and Kurosawa-Desmedt [25] with explicit rejection. This yields KEMs with a 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 KEM, the security proofs do not immediately extend to the modified KEMs. Hence, it is not obvious what level of security these schemes provide.

1.1 Our Contribution

In this paper, we discuss the impossibility of proving CCA security of KEMs in standard prime order groups with low ciphertext overhead. More specifically, as our main result, we show that there is no algebraic black-box reduction from the q-bounded one-way non-adaptive CCA security ($\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}$) 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 [18, 26]. These results imply that the approach of minimizing ciphertext overhead in the above mentioned schemes [20, 15, 25], 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. [9] is contained in the KEM class, our results imply that this scheme cannot be shown fully CCA secure or non-malleable based on any non-interactive assumption.

Secondly, we show a simple construction of a CCA-secure KEM using programmable hash functions introduced by Hofheinz and Kiltz [21]. These hash functions capture the "programmability" achieved by a random oracle, and have been shown useful, for example, in the construction of short signatures in the standard model [21, 19]. We show that an algebraic $(q, 1)$-programmable hash function allows the construction of a q-bounded CCA ($\mathtt{IND}\text{-}q\text{-}\mathtt{CCA2}$) secure KEM based on the DDH assumption, 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 $\beta > 0$. This result answers an open question posed by Hofheinz and Kiltz [21] in the case of algebraic programmable hash functions in prime order groups. We note that all known constructions of programmable hash functions are algebraic [21, 19], and that the properties of these hash functions suggest that this may be inherent [21, 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* [23]. 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 [23, 28] and meta-reductions [3, 8]. The former technique is typically used to show separations between primitives, e.g. [23, 29, 12, 14, 13], 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 e.g. [3, 8, 1]. For an overview of these definitions, techniques, and results, see [28, 32].

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 the 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 an oracle is sufficient to rule out a fully black-box reduction [28] 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 [3, 8, 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 are 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 [3, 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 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, and if $\mathbb{D}$ is a probability distribution, then "$x \leftarrow \mathbb{D}$" denotes choosing x according to $\mathbb{D}$. 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 $f(\lambda) : \mathbb{N} \to [0, 1]$ 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 $\mathbf{X}$.

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 λ and the group description Λ generated by $\mathsf{GS}(1^\lambda)$.

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 $\mathbf{X}$.

The above definition of algebraic algorithms can naturally be extended to algebraic "oracle" algorithms, which play an important role in our result. Specifically, let $\mathcal{R}$ be an oracle PPTA that takes $\Lambda = (g, p, \mathbb{G})$ (output by $\mathsf{GS}(1^\lambda)$), 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 $q = q(\lambda)$ 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 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.

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 consisting of a problem instance and a witness (y, w).

Verification algorithm V: This algorithm takes a problem instance y, 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 $\mathsf{V}(y, x, w) = \top$, 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:

$$(g, p, \mathbb{G}) \leftarrow \mathsf{GS}(1^\lambda);\ (y, w) \leftarrow \mathsf{I}(\Lambda);\ x \leftarrow \mathcal{A}(1^\lambda, y);\ \text{Return } (\mathsf{V}(y, x, w) \overset{?}{=} \top)$$

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 3 *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* $\mathsf{Adv}^{\mathsf{P}}_{\mathsf{GS},\mathcal{A}}(\lambda) \leq \mu(\lambda)$*.*

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, q-ABDHE, etc.), and will even capture the CPA security of a KEM.

Key Encapsulation. 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, and change the security experiments accordingly.

A KEM $\mathit{\Gamma}$ with respect to a group scheme GS consists of the following three PPTAs $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$. KG is a key generation algorithm which 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); Enc is an encapsulation algorithm which 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); and Dec is a decapsulation algorithm which takes sk and c as input, and outputs a session-key K or an error symbol $\perp$ indicating that c is "invalid". We require $\mathsf{Dec}(sk, c) = K$ for all (pk, sk) output by $\mathsf{KG}(\mathit{\Lambda})$ and all (c, K) output by $\mathsf{Enc}(pk)$.

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 consider *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 [9] (q-`CCA1` and q-`CCA2`) as an adversary's attack types `ATK`. Due to space limitations, we will only define `OW`-q-`CCA1` security. See the full version for the remaining definitions.

We say that a KEM $\mathit{\Gamma} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ with respect to GS is `OW`-q-`CCA1` secure if, for any PPTA $\mathcal{A}$, the following advantage function is negligible in λ:

$$\begin{aligned}\mathsf{Adv}^{\mathtt{OW}\text{-}q\text{-}\mathtt{CCA1}}_{\mathsf{GS},\mathcal{A}}(\lambda) = \Pr[&\mathit{\Lambda} \leftarrow \mathsf{GS}(1^{\lambda});\ (pk, sk) \leftarrow \mathsf{KG}(\mathit{\Lambda});\ \mathsf{st} \leftarrow \mathcal{A}_1^{\mathsf{Dec}(sk,\cdot)}(pk);\\ &(c^*, K^*) \leftarrow \mathsf{Enc}(pk);\ K' \leftarrow \mathcal{A}_2(\mathsf{st}, c^*) : K^* = K']\end{aligned}$$

where the decapsulation oracle $\mathsf{Dec}(sk, \cdot)$ can only be queried q times.

Algebraic Black-Box Reductions. We will consider the following type of reductions from the security of a KEM to the hardness of a non-interactive problem.

Definition 4 *Let* GS *be a group scheme, and let* Γ *be a KEM and* P *be a NIP with respect to* GS*. Furthermore, let* $\mathtt{GOAL}\text{-}\mathtt{ATK}$ *be a security notion for a KEM. We say that there is an* algebraic black-box reduction *from the* $\mathtt{GOAL}\text{-}\mathtt{ATK}$ *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* $\mathsf{Adv}^{\mathtt{GOAL}\text{-}\mathtt{ATK}}_{\Gamma,\mathcal{A}}(\lambda)$ *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. [28]. In particular, the 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 [10], Kurosawa-Desmedt [25] (with explicit rejection), Hofheinz-Kiltz [20], Hanaoka-Kurosawa [15], and Cramer et al. [9], 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 [30] (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 5 *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* $pk = (\Lambda, aux, X_1, \ldots, X_n)$*.*
2. *A ciphertext* $C \in \mathbb{G} \times \{0,1\}^*$ *and the corresponding session-key* $K \in \mathbb{G}$ *are of the form*

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

where $r \leftarrow \mathbb{Z}_p$*, and* $\widetilde{f} : \mathcal{PK} \times \mathbb{Z}_p \to \{0,1\}^*$ *and* $f_0, \ldots, f_n : \mathcal{PK} \times \mathbb{Z}_p \to \mathbb{Z}_p$ *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.
– 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 [5].

4 Main Impossibility Result

The following theorem captures our main result.

Theorem 6 *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 [23, 28]. Specifically, the theorem follows from the following lemma. (In the lemma, $\mathsf{Adv}^{\mathtt{DL}}_{\mathsf{GS},\mathcal{A}}(\lambda)$ denotes the advantage of an algorithm $\mathcal{A}$ in solving the discrete logarithm problem wrt. GS. See the full version for a formal definition.)

Lemma 7 *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 an oracle* $\mathcal{O}$ *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*

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

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. (A formal proof can be found in the full version.) 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 $\mathtt{GOAL}\text{-}\mathtt{ATK}$ security of the KEM Γ to P. The latter guarantees that there exists an algebraic oracle PPTA $\mathcal{R}$ that satisfies Definition 4. 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 $\mathtt{GOAL}\text{-}\mathtt{ATK}$ 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 $\mathtt{GOAL}\text{-}\mathtt{ATK}$ 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 $\mathtt{GOAL}\text{-}\mathtt{ATK}$ 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 $\mathit{\Gamma}$ to the hardness of P. $\square$

To make use of the above Lemma 7, 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 6.

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 $\mathit{\Gamma} \in \mathcal{K}_{\mathsf{GS},n}$ 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 $\mathit{\Gamma}$ (i.e. $pk = (\mathit{\Lambda}, aux, X_1, \ldots, X_n)$ and $\mathit{\Lambda} = (g, p, \mathbb{G}) \leftarrow \mathsf{GS}(1^\lambda)$):

- $\mathcal{O}_1$ 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 $i = [n]$, 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 $\mathit{\Gamma}$.

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 defined oracle $\mathcal{O}$. 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 $\mathcal{K}_{\mathsf{GS},n}$.

Lemma 8 *Let* GS *be a group scheme,* $\mathit{\Gamma} \in \mathcal{K}_{\mathsf{GS},n}$ *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* $\mathbf{E}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\mathit{\Gamma},\mathcal{A}^{\mathcal{O}}}(\lambda)] \geq \frac{3}{4n^2}$.

Proof Sketch. (A formal proof can be found in the full version.) In the following, we consider the $\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}$ 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 is given by $\mathbf{E}_{\mathcal{O}\leftarrow\mathbb{D}}[\mathsf{Adv}^{\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}}_{\mathit{\Gamma},\mathcal{A}^{\mathcal{O}}}(\lambda)]$.

Recall that the session-key K of a ciphertext $C = (c, d)$ of the KEM $\mathit{\Gamma}$ 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 $\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}$ 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 $\mathcal{O}$ and the decapsulation oracle $\mathcal{O}_{dec}$ to calculate this component for the challenge ciphertext C^*, and thereby break the $\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}$ security of $\mathit{\Gamma}$. $\mathcal{A}$ is constructed as follows:

Given a public key $pk = (\mathit{\Lambda}, aux, X_1, \ldots, X_n)$ for $\mathit{\Gamma}$, $\mathcal{A}$ simply submits this to $\mathcal{O}_1$ to obtain n randomly generated ciphertexts $(C_1, \ldots, C_n)$ under pk. Then $\mathcal{A}$ submits $(C_1, \ldots, C_n)$ to $\mathcal{O}_{dec}$ to obtain the corresponding decapsulations $(K_1, \ldots, K_n)$. Lastly, $\mathcal{A}$ will submit $(K_1, \ldots, K_n)$ 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 $j \in [n]$. 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 trivally calculate K^*.

The tricky part is the case in which the vectors $\{\boldsymbol{\psi}(pk, C_j)\}_{j\in[j]}$ 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 $\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}$ 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. □

Breaking Non-malleability. Inspecting the proof of the above lemma reveals that the n decapsulation queries made by $\mathcal{A}$ can be done in a single parallel decapsulation query containing n ciphertexts. Since indistinguishability against a parallel chosen ciphertext attack is implied by the notion of non-malleability [18, 26], 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 oracle defined in Section 4.1 is essentially useless for an algebraic algorithm trying to solve a NIP.

Lemma 9 *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. (A formal proof can be found in the full version.) 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 $(pk, K_1, \dots, K_n, \sigma)$ (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 $(u_1, \dots, u_n)$. 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 $\psi_1(pk, C_i, y_1, \dots, y_n)$. 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. $\square$

5 Lower Bounds 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 the DDH assumption, 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 10 *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 $s_1, \dots, s_\alpha \in \{0,1\}^{\ell(\lambda)}$ and $s'_1, \dots, s'_\beta \in \{0,1\}^{\ell(\lambda)}$ such that $s_i \neq s'_j$ for all i, j, the probability $\Pr[a_{s_1} = \cdots = a_{s_\alpha} = 0 \wedge a_{s'_1}, \dots, a_{s'_\beta} \neq 0]$ 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.

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 [21, 19] are algebraic.

5.2 A Simple KEM Based on a Programmable Hash Function

We now show how to construct an IND-q-CCA2 secure KEM with ciphertexts consisting of 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. Then we construct a KEM Γ as follows:

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. [9] can be seen as a concrete instantiation of the above KEM in which we use the concrete programmable hash function proposed in [19, Sect. 3.3].

Theorem 11 *Let* GS *be a group scheme. 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 the full version.

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. Hence Theorem 6 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[4]. On the other hand, the above Theorem 11 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 12 *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 [21] in the case of algebraic programmable hash function defined in prime order groups.

Theorem 13 *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 [10, 20], [15, Sect. 4.1], and [25] (with explicit rejection), but requires the ciphertext to consist of just a single random group element and a string.

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 [20] secure KEMs based on non-interactive assumptions.

Furthermore, we have focused on simple KEMs in which the session-key lies within the group. More precisely, the KEM class $\mathcal{K}_{\mathsf{GS},n}$ does not capture the

[4] Note that this does not contradict the results by Cramer et al. [9]. More specifically, while the KEM defined in [9] 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.

structure of schemes which make use of a pairing to derive the session-key like [4], or apply a type of key-derivation function, such as the hardcore bit-based schemes like [15, Sect. 5], [17, Sect. 3], and [31, Sect. 3], the HDH-based versions of [15, Sect. 4.2] and [6, Sect. 6.2], or a combination of these like [17, Sect. 5.2 and 5.3] and [31, 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 $\mathtt{OW}\text{-}n\text{-}\mathtt{CCA1}$ secure based on a non-interactive assumption 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 [22, 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] 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. Furthermore, the definition of a programmable hash function requires the hash function to have some "algebraic properties" (see the discussion in [21, Sect. 1.5]), which seems to suggest that 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, 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*, pp. 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*, pp. 358–374, 2008.
3. D. Boneh and R. Venkatesan. Breaking rsa may not be equivalent to factoring. In *EUROCRYPT*, pp. 59–71, 1998.

4. X. Boyen, Q. Mei, and B. Waters. Direct chosen ciphertext security from identity-based techniques. Cryptology ePrint Archive, Report 2005/288, 2005.
5. J. Camenisch, K. Haralambiev, M. Kohlweiss, J. Lapon, and V. Naessens. Structure preserving cca secure encryption and applications. In *ASIACRYPT*, pp. 89–106, 2011.
6. 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 [7].
7. D. Cash, E. Kiltz, and V. Shoup. The twin diffie-hellman problem and applications. In *EUROCRYPT*, pp. 127–145, 2008.
8. J.-S. Coron. Optimal security proofs for pss and other signature schemes. In *EUROCRYPT*, pp. 272–287, 2002.
9. R. Cramer, G. Hanaoka, D. Hofheinz, H. Imai, E. Kiltz, R. Pass, A. Shelat, and V. Vaikuntanathan. Bounded cca2-secure encryption. In *ASIACRYPT*, pp. 502–518, 2007.
10. 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.
11. K. Emura, G. Hanaoka, T. Matsuda, G. Ohtake, and S. Yamada. Chosen ciphertext secure keyed-homomorphic public-key encryption, 2012. Manuscript.
12. Y. Gertner, S. Kannan, T. Malkin, O. Reingold, and M. Viswanathan. The relationship between public key encryption and oblivious transfer. In *FOCS*, pp. 325–335, 2000.
13. Y. Gertner, T. Malkin, and S. Myers. Towards a separation of semantic and cca security for public key encryption. In *TCC*, pp. 434–455, 2007.
14. Y. Gertner, T. Malkin, and O. Reingold. On the impossibility of basing trapdoor functions on trapdoor predicates. In *FOCS*, pp. 126–135, 2001.
15. 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 [16].
16. G. Hanaoka and K. Kurosawa. Efficient chosen ciphertext secure public key encryption under the computational diffie-hellman assumption. In *ASIACRYPT*, pp. 308–325, 2008.
17. K. Haralambiev, T. Jager, E. Kiltz, and V. Shoup. Simple and efficient public-key encryption from computational diffie-hellman in the standard model. In *PKC*, pp. 1–18, 2010.
18. J. Herranz, D. Hofheinz, and E. Kiltz. Some (in)sufficient conditions for secure hybrid encryption. *Inf. Comput.*, 208(11):1243–1257, 2010.
19. D. Hofheinz, T. Jager, and E. Kiltz. Short signatures from weaker assumptions. In *ASIACRYPT*, pp. 647–666, 2011.
20. D. Hofheinz and E. Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO*, pp. 553–571, 2007.
21. D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. In *J. of Cryptology*, 25(3):484–527, 2012.
22. D. Hofheinz and E. Kiltz. The group of signed quadratic residues and applications. In *CRYPTO*, pp. 637–653, 2009.
23. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *STOC*, pp. 44–61, 1989.
24. E. Kiltz. Chosen-ciphertext secure key-encapsulation based on gap hashed diffie-hellman. In *PKC*, pp. 282–297, 2007.

25. K. Kurosawa and Y. Desmedt. A new paradigm of hybrid encryption scheme. In *CRYPTO*, pp. 426–442, 2004.
26. T. Matsuda and K. Matsuura. Parallel decryption queries in bounded chosen ciphertext attacks. In *Public Key Cryptography*, pp. 246–264, 2011.
27. D. H. Phan and D. Pointcheval. About the security of ciphers (semantic security and pseudo-random permutations). In *SAC*, pp. 182–197, 2004.
28. O. Reingold, L. Trevisan, and S. P. Vadhan. Notions of reducibility between cryptographic primitives. In *TCC*, pp. 1–20, 2004.
29. D. R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In *EUROCRYPT*, pp. 334–345, 1998.
30. B. Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT 2005*, pp. 114–127, 2005.
31. 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.
32. 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.