On the Necessity of Strong Assumptions for the Security of a Class of Asymmetric Encryption Schemes

Ron Steinfeld[1], Joonsang Baek[1], and Yuliang Zheng[2]

[1] School of Network Computing, Monash University, McMahons Road, Frankston, VIC 3199, Australia
{joonsang.baek,ron.steinfeld}@infotech.monash.edu.au
[2] Dept. Software and Info. Systems, UNC Charlotte, NC 28223, USA
yzheng@uncc.edu

Abstract. Recently various public key encryption schemes such as DHIES by Abdalla, Bellare and Rogaway and REACT by Okamoto and Pointcheval, whose security against adaptive chosen ciphertext attack (CCA) is based on the Gap problems, have been proposed. Although the Gap problems were proved to be a sufficient assumption for those schemes to be secure against adaptive chosen-cipertext attack, a necessary condition for CCA security of those schemes has not been explicitly discussed.
In this paper we clarify the necessary condition for CCA security of those schemes. Namely we prove (in the random oracle model) that the Gap Diffie-Hellman is not only sufficient, but also a *necessary* assumption for the CCA security of DHIES and Diffie-Hellman version of REACT. We also show that our result applies to a wider class of public key encryption schemes. Furthermore we show that our result implies the equivalence, in the random oracle model, between 'Strong Diffie-Hellman' and 'Oracle Diffie-Hellman' assumptions proposed by Abdalla, Bellare and Rogaway. Our results may be used as criteria for distinguishing public key encryption schemes whose CCA security is based on strong assumptions (such as Gap Diffie-Hellman) from those schemes based on weaker ones (such as Computational Diffie-Hellman).

1 Introduction

The design of practical public-key encryption schemes which are secure against powerful attacks, namely adaptive chosen ciphertext attacks, has been a very active research topic since the early work of Zheng and Seberry [6]. Recently, a simple and efficient public key encryption scheme called 'Diffie-Hellman Integrated Encryption Scheme (DHIES)', was proposed and analysed by Abdalla Bellare and Rogaway [4]. In [4], the authors state that due to technical problems, it seems hard to prove that the scheme DHIES is secure against adaptive Chosen Ciphertext Attack (CCA) assuming only the standard 'Computational Diffie-Hellman' (CDH) assumption holds in the underlying group, and the random oracle model [2] for the hash function used in the scheme. Instead, the authors proved that the DHIES scheme is secure against adaptive chosen ciphertext attack in the random oracle model with respect to a strong computational assumption on the underlying group called 'Strong Diffie-Hellman' (SDH) and even secure in the standard model (i.e., not assuming the underlying hash function behaves as a random oracle) if another strong assumption called 'Oracle Diffie-Hellman (ODH)' is considered.

In this paper we clarify the technical problems in proving that DHIES is secure against adaptive chosen-ciphertext attack in the random oracle model assuming the CDH assumption. We present a simple chosen ciphertext attack which efficiently breaks DHIES if the SDH assumption does not hold. That is, we show that SDH is not only sufficient, but also a *necessary* assumption for the CCA security of DHIES. Our attack shows that the technical difficulties in proving DHIES is as secure as the CDH assumption cannot be overcome by a more careful analysis of the scheme DHIES. Rather, the problem is purely a computational one in the underlying group, namely to prove that SDH is equivalent to CDH. As an application of our work we also clarify the relationship between the two new assumptions introduced by the authors of [4], 'Oracle Diffie-Hellman' (ODH) and 'Strong Diffie-Hellman' (SDH). We show that these two assumptions are in fact equivalent in the random oracle model.

As an extension of our work we consider Okamoto and Pointcheval' recent scheme called 'Rapid Enhanced-Security Asymmetric Cryptosystem Transform' (REACT) [5]. This transformation converts any 'weakly secure' encryption scheme (namely a 'One-Way Plaintext Checking Attacker', or OW-PCA) scheme) into an encryption scheme secure against adaptive chosen-cipertext attack in the random oracle model. We show that a variation of our attack breaks REACT in the CCA sense if the OW-PCA assumption does not hold, i.e. OW-PCA is necessary as well as sufficient for the CCA security of REACT.

To emphasize the generality of our attack we present it in the following way. We consider a primitive called a 'Key Encapsulation Mechanism (KEM)' [7] inspired by the Diffie-Hellman function, and propose a corresponding 'OW-PCA' notion for it. This primitive can be simpler than a 'weakly secure encryption scheme', since it only needs to generate a uniformly distributed key and a ciphertext for it, rather than encrypting specified input messages. Then we describe two variants 'CCAKEM1' and 'CCAKEM2' of a conversion from any KEM to an encryption scheme secure against adaptive chosen-ciphertext attack in the random oracle model. The first variant CCAKEM1 uses one random oracle and one Message Authentication Code (MAC) and includes DHIES as a special instance in which the KEM is the Diffie-Hellman one. The second variant CCAKEM2 uses two random oracles and includes REACT as a subclass in which the KEM is implemented using an encryption scheme. We present variations of our attack for both CCAKEM1 and CCAKEM2 if the 'OW-PCA' assumption on the KEM does not hold. This implies as special cases the above-mentioned attacks on DHIES and REACT.

2 Preliminaries

In this section we review the KEM, and define a security notion for it called 'One-Wayness under Plaintext Checking Attacks' (OW-PCA). This notion is analogous to the OW-PCA notion defined for encryption schemes by Okamoto and Pointcheval in [5]. Note that definitions of asymmetric encryption schemes, symmetric encryption schemes, and MAC and the standard security notions for them are given in the Appendix.

2.1 Notation

We use the notation $\mathsf{A}(.,.)$ to denote an algorithm, with input arguments separated by commas (our underlying computational model is a probabilistic Turing Machine). If algorithm A makes calls to oracles, we list the oracles separated from the algorithm inputs by the symbol '$|$'. Given a set SP_{sk} we denote by $sk \overset{\mathrm{R}}{\leftarrow} SP_{sk}$ the assignment of a uniformly and independently distributed random element from the set SP_{sk} to the variable sk. Given an element $h \in \{0,1\}^k$, we denote by $h[i,...,j]$ the substring of h consisting of the bits at positions i to j (where bit 1 is by convention the rightmost bit). We use the notation $\Pr[\mathsf{Event}]_{exp}$ to denote the probability of event Event in experiment exp.

2.2 Key Encapsulation Mechanism (KEM)

The KEM is defined as follows.

Definition 1. *A Key Encapsulation Mechanism (KEM) consists of 3 algorithms:*

1 *Key-Pair Generation Algorithm* $\mathsf{GK}(k)$ *— Takes a security parameter* $k \in \mathbb{N}$ *and generates a secret and public key pair* (sk, pk).
2 *Random Key Encryption Algorithm* $\mathsf{E}^{\mathsf{KEM}}_{\mathsf{pk}}(r)$ *— Takes a recipient's public key* pk *and a random string* $r \in SP_R$*, and outputs a pair* (K, c)*, where* $K \in SP_K$ *is a key and* c *is a ciphertext for* K.
3 *Random Key Decryption Algorithm* $\mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c)$ *— Takes a recipient's secret key* sk *and a ciphertext* c *and outputs a decrypted key* K.

We require that for every key pair (sk, pk) *output by* $\mathsf{GK}(k)$ *and each* $r \in SP_R$*, it is the case that if* $(K, c) = \mathsf{E}^{\mathsf{KEM}}_{\mathsf{pk}}(r)$ *then* $\mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c) = K$.

Example. The Diffie-Hellman KEM (DHKEM) in a multiplicatively-written group G is described as follows.

1 Algorithm $\mathsf{GK}(k)$ outputs common parameters (d_G, g, q) consisting of description d_G of a finite cyclic group G, a generator $g \in G$ and the order q of G and chooses $x \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q$ and computes $y \leftarrow g^x$ in G. It outputs (sk, pk), where $sk = x$ and $pk = (d_G, g, q, y)$.
2 Algorithm $\mathsf{E}^{\mathsf{DHKEM}}_{\mathsf{pk}}(r)$ accepts $r \overset{\mathrm{R}}{\leftarrow} \mathbb{Z}_q$, computes key $K \leftarrow y^r$ and ciphertext $c \leftarrow g^r$ and outputs (K, c).

3 Algorithm $\mathsf{D}_{\mathsf{sk}}^{\mathsf{DHKEM}}(c)$ computes $K \leftarrow c^x$ and outputs K.

Notice that the DHKEM is simpler than the Diffie-Hellman-based El-Gamal encryption scheme because in KEM there is no need to encrypt a specified input message, only to encrypt a key derived from the input random string. Of course, any public-key encryption scheme can also function as an KEM by setting $K = r$, where r is a random message, and c is the encryption of r.

Analogously to [5], we define the 'OW-PCA security notion for an KEM as follows. First a plaintext checking oracle is defined.

Definition 2. (Plaintext Checking Oracle) *Given a Key Encapsulation Mechanism* $\mathsf{KEM} = (\mathsf{GK}, \mathsf{E}^{\mathsf{KEM}}, \mathsf{D}^{\mathsf{KEM}})$ *and a key pair* (sk, pk) *output by* GK*, we define an associated* Plaintext Checking Oracle *(PCO) algorithm* $\mathsf{PCO}_{\mathsf{KEM},\mathsf{sk}}(.,.)$*, where given a key* $K \in SP_K$ *and a ciphertext* c*,* $\mathsf{PCO}_{\mathsf{KEM},\mathsf{sk}}(K, c)$ *returns 1 if* $\mathsf{D}_{\mathsf{sk}}^{\mathsf{KEM}}(c) = K$ *and else returns 0.*

Then the OW-PCA notion is defined in a quantitative way.

Definition 3. (OW-PCA) *Let* $\mathsf{KEM} = (\mathsf{GK}, \mathsf{E}^{\mathsf{KEM}}, \mathsf{D}^{\mathsf{KEM}})$ *be a KEM. Let* A *be an attack algorithm. Define the experiment*

```
Experiment $\mathbf{OWPCAExp}(k, \mathsf{KEM}, \mathsf{A})$
    $(sk, pk) \leftarrow \mathsf{GK}(k)$
    $r \overset{\mathrm{R}}{\leftarrow} SP_R$; $(K, c) \leftarrow \mathsf{E}_{\mathsf{pk}}^{\mathsf{KEM}}(r)$
    $K' \leftarrow \mathsf{A}(pk, c|\mathsf{PCO}_{\mathsf{KEM},\mathsf{sk}}(.,.))$
    If $K' = K$ then Return 1 else Return 0
```

We quantify A*'s success in breaking the OW-PCA notion of* KEM *by the probability* $\mathbf{Succ}_{\mathsf{A},\mathsf{KEM}}^{\mathrm{OW-PCA}}(k) \overset{\mathrm{def}}{=} \Pr[\mathbf{OWPCAExp}(k, \mathsf{KEM}, \mathsf{A}) = 1]$*. We define* A*'s resource parameters as* $RP = (t, q_{PC})$ *if* A *has running time/program size at most* t *and makes at most* q_{PC} *queries to the PCO oracle.*

Note that the attacker A is allowed to query the part of challenge ciphertext c.

3 The Transforms 'CCAKEM1' and 'CCAKEM2'

In this section we define the two transformations schemes 'CCAKEM1' and 'CCAKEM2' which convert any OW-CPA key encapsulation mechanism into an asymmetric encryption scheme secure against adaptive chosen ciphertext attack in the random oracle model, and explain how REACT and DHIES are related to them.

3.1 Transform CCAKEM1

The transform 'CCAKEM1' takes (1) A key encapsulation mechanism $\mathsf{KEM} = (\mathsf{GK}^{\mathsf{KEM}}, \mathsf{E}^{\mathsf{KEM}}, \mathsf{D}^{\mathsf{KEM}})$, (2) A MAC $\mathsf{MAC} = (\mathsf{MACG}, \mathsf{MACV})$ with key space $\{0,1\}^{l_m}$, (3) An IND-CPA symmetric encryption scheme $\mathsf{SYM} = (\mathsf{E}^{\mathsf{SYM}}, \mathsf{D}^{\mathsf{SYM}})$ with key space $\{0,1\}^{l_e}$, and (4) A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^{l_e+l_m}$, modelled as a random oracle [2], and constructs an asymmetric encryption scheme $\mathsf{CCAKEM1} = (\mathsf{GK}^{\mathsf{CCAKEM1}}, \mathsf{E}^{\mathsf{CCAKEM1}}, \mathsf{D}^{\mathsf{CCAKEM1}})$ as follows.

Transform **CCAKEM1**

```
$\mathsf{GK}^{\mathsf{CCAKEM1}}(k)$
    $(sk, pk) \leftarrow \mathsf{GK}^{\mathsf{KEM}}(k)$
    Return $(sk, pk)$

$\mathsf{E}_{\mathsf{pk}}^{\mathsf{CCAKEM1}}(m)$
    $r \overset{\mathrm{R}}{\leftarrow} SP_R$
    $(K, c) \leftarrow \mathsf{E}_{\mathsf{pk}}^{\mathsf{KEM}}(r)$
    $h \leftarrow H(K, c)$
    $km \leftarrow h[1, ..., l_m]$
    $ke \leftarrow h[l_m + 1, ..$
    $.., l_m + l_e]$
    $c_s \leftarrow \mathsf{E}_{ke}^{\mathsf{SYM}}(m)$
    $\sigma_s \leftarrow \mathsf{MACG}_{km}(c_s)$
    Return $(c, c_s, \sigma_s)$

$\mathsf{D}_{\mathsf{sk}}^{\mathsf{CCAKEM1}}((c, c_s, \sigma_s))$
    $K \leftarrow \mathsf{D}_{\mathsf{sk}}^{\mathsf{KEM}}(c)$
    $h \leftarrow H(K, c)$
    $km \leftarrow h[1, ..., l_m]$
    $ke \leftarrow h[l_m + 1, ..., l_m + l_e]$
    $m \leftarrow \mathsf{D}_{ke}^{\mathsf{SYM}}(c_s)$
    $d \leftarrow \mathsf{MACV}_{km}(c_s, \sigma_s)$
    If $d = Acc$ Return $m$
    Else Return $Rej$
```

Observe that this scheme is a very natural one for performing 'hybrid encryption'. To encrypt a message m, one first uses the KEM to encapsulate a 'session key' K into a KEM ciphertext c. Then the session key (and ciphertext c) is hashed using $H(.)$ to derive two symmetric keys: one key ke is used encrypt m into a ciphertext c_s using the symmetric encryption scheme and another key km is used to generate a MAC tag σ_s on the symmetric ciphertext c_s using the MAC scheme. The decryption algorithm recovers the session key K and then the symmetric keys, checking the MAC tag for validity before decrypting c_s to recover m.

We remark that by setting the scheme KEM to be the Diffie-Hellman KEM described in the previous section, this transformation yields the DHIES scheme, and the OW-PCA assumption on the KEM becomes the 'Strong Diffie-Hellman' Assumption (SDH): given (g^a, g^b), compute g^{ab} given a fixed-input Decision Diffie-Hellman (DDH) oracle, which given a pair of group elements (y, z) decides whether $z = y^a$ or not.

3.2 Transform CCAKEM2

The transform 'CCAKEM2' takes (1) A key encapsulation mechanism $\mathsf{KEM} = (\mathsf{GK}^{\mathsf{ARK}}, \mathsf{E}^{\mathsf{KEM}}, \mathsf{D}^{\mathsf{KEM}})$, (2) An IND-CPA-secure Symmetric encryption scheme $\mathsf{SYM} = (\mathsf{E}^{\mathsf{SYM}}, \mathsf{D}^{\mathsf{SYM}})$ with key space $\{0,1\}^{l_e}$, and (3) Two hash functions $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{l_e}$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^{l_\sigma}$, both modeled as random oracles, and constructs an asymmetric encryption scheme $\mathsf{CCAKEM2} = (\mathsf{GK}^{\mathsf{CCAKEM2}}, \mathsf{E}^{\mathsf{CCAKEM2}}, \mathsf{D}^{\mathsf{CCAKEM2}})$ as follows.

Transform **CCAKEM2**

```
$\mathsf{GK}^{\mathsf{CCAKEM2}}(k)$
    $(sk, pk) \leftarrow \mathsf{GK}^{\mathsf{KEM}}(k)$
    Return $(sk, pk)$

$\mathsf{E}_{\mathsf{pk}}^{\mathsf{CCAKEM2}}(m)$
    $r \overset{\mathrm{R}}{\leftarrow} SP_R$; $(K, c) \leftarrow \mathsf{E}_{\mathsf{pk}}^{\mathsf{KEM}}(r)$
    $ke \leftarrow H_1(K, c)$
    $c_s \leftarrow \mathsf{E}_{\mathsf{ke}}^{\mathsf{SYM}}(m)$
    $\sigma_s \leftarrow H_2(K, m, c, c_s)$
    Return $(c, c_s, \sigma_s)$

$\mathsf{D}_{\mathsf{sk}}^{\mathsf{CCAKEM2}}((c, c_s, \sigma_s))$
    $K \leftarrow \mathsf{D}_{\mathsf{sk}}^{\mathsf{KEM}}(c)$
    $ke \leftarrow H_1(K, c)$
    $m \leftarrow \mathsf{D}_{\mathsf{ke}}^{\mathsf{SYM}}(c_s)$
    $d \leftarrow H_2(K, m, c, c_s)$
    If $d = \sigma_s$ Return $m$
    Else Return $Rej$
```

This scheme uses a similar natural approach as the previous one but using a second hash function for tagging the ciphertext. To encrypt a message m, one first uses the KEM to encapsulate a 'session key' K into a KEM ciphertext c. Then the session key is hashed using $H_1(.)$ to derive a symmetric encryption key ke used encrypt m into a ciphertext c_s using the symmetric encryption scheme. Then a tag σ_s is also generated using hash function $H_2(.)$ by hashing all of K, m, c, c_s. The decryption algorithm recovers the session key K and then the symmetric key ke and the decrypted message m, checking hash tag σ_s for validity before returning m.

We observe that by implementing the scheme KEM using a public key encryption scheme, namely by choosing a random message element as the key K and encrypting it, we obtain the REACT transformation. The OW-PCA assumption on the KEM becomes the OW-PCA assumption on the encryption scheme. As observed in [5], the OW-PCA assumption is equivalent to just the one-wayness assumption in the case of a deterministic encryption scheme with a one-to-one decryption algorithm (such as RSA), since the PC oracle can be implemented by re-encryption.

4 The Attacks

Now we present our attacks on the above conversions, assuming that the underlying KEM is *not* OW-PCA.

4.1 Attack on CCAKEM1 if the underlying KEM is not OW-PCA

Theorem 1. *Let* A^{PC} *be an attack algorithm with resource parameters* (t^{PC}, q_{PC}) *for breaking OW-PCA of the key encapsulation mechanism* KEM. *Then we can construct an attack algorithm* $\mathsf{A}^{\mathsf{CC}} = (\mathsf{A}^{\mathsf{CC}}_{\mathsf{find}}, \mathsf{A}^{\mathsf{CC}}_{\mathsf{guess}})$ *with resource parameters* (t^{CC}, q_D, q_H) *such that*

$$\mathbf{Succ}^{\mathrm{CCA}}_{\mathsf{A}^{\mathsf{CC}},\mathsf{CCAKEM1}}(k) \geq \mathbf{Succ}^{\mathrm{OW-PCA}}_{\mathsf{A},\mathsf{KEM}}(k) - 2(q_{PC}+2)\mathbf{InSec}^{\mathrm{MAC-UF}}_{\mathsf{MAC}}(t^{CC}, 0, q_{PC})$$

and $t^{CC} = t^{PC} + (q_{PC}+1)(O(1) + t_{MAC})$, $q_D = q_{PC}$, $q_H = 2q_{PC}+1$ *(here* t_{MAC} *denotes the time to evaluate* MACG *or* MACV*).*

Proof. We construct CCA attacker A^{CC}. The idea of the construction is simple — A^{CC} essentially runs A^{PC} on the OW-PCA instance of KEM corresponding to the challenge ciphertext given to A^{CC}, and simulates the $\mathsf{PCO}_{\mathsf{KEM},\mathsf{sk}}(.,.)$ oracle to which A^{PC} makes queries, using the decryption oracle $\mathsf{D}^{\mathsf{CCAKEM1}}_{\mathsf{sk}}(.)$. Although the PC oracle simulation is not perfect, we will bound its error probability using the assumed bound on the insecurity of the MAC scheme.

We first give a detailed definition of the two sub-attacker algorithms $\mathsf{A}^{\mathsf{CC}}_{\mathsf{find}}$ and $\mathsf{A}^{\mathsf{CC}}_{\mathsf{guess}}$ making up A, as well as the PCO simulator algorithm $\mathsf{PCOSim}(.,.)$ which is used to answer the PCO queries of A^{PC} when the latter is run by $\mathsf{A}^{\mathsf{CC}}_{\mathsf{guess}}$. Then we analyse the attack to prove the claims of the theorem.

CCA Attacker A^{CC} Against Scheme **CCAKEM1**

```
$\mathsf{A}^{\mathsf{CC}}_{\mathsf{find}}(pk|\mathsf{D}^{\mathsf{CCAKEM1}}_{\mathsf{sk}}(.))$ (Find Stage)
    Let $m_0$ and $m_1$ denote distinct messages in $SP^{SYM}_M$.
    $s \leftarrow (m_0, m_1)$
    Return $(m_0, m_1, s)$
$\mathsf{A}^{\mathsf{CC}}_{\mathsf{guess}}(pk, m_0, m_1, s, (c, c_s, \sigma_s)|\mathsf{D}^{\mathsf{CCAKEM1}}_{\mathsf{sk}}(.))$ (Guess Stage)
    $K' \leftarrow \mathsf{A}^{\mathsf{PC}}(pk, c|\mathsf{PCOSim}(., .))$
    $h' \leftarrow H(K', c)$
    $km' \leftarrow h'[1, ..., l_m]$; $ke' \leftarrow h'[l_m + 1, ..., l_m + l_e]$
    If $d' \overset{\text{def}}{=} \mathsf{MACV}_{km'}(c_s, \sigma_s) =$ '$Rej$' then $b' \overset{\text{R}}{\leftarrow} \{0, 1\}$
    Else
        $m' \leftarrow \mathsf{D}^{\mathsf{SYM}}_{ke'}(c_s)$
        If $m' = m_j$ for $j \in \{0, 1\}$ then $b' \leftarrow j$ Else $b' \overset{\text{R}}{\leftarrow} \{0, 1\}$
    Return $b'$
$\mathsf{PCOSim}(K[i], c[i]|H(.), \mathsf{D}^{\mathsf{CCAKEM1}}_{\mathsf{sk}}(.))$ (PCO Simulator)
    $h[i] \leftarrow H(K[i], c[i])$
    $km[i] \leftarrow h[i][1, ..., l_m]$; $ke[i] \leftarrow h[l_m + 1, ..., l_m + l_e]$
    Find $j \in \{0, 1\}$ such that $c_s[i] \overset{\text{def}}{=} \mathsf{E}^{\mathsf{SYM}}_{ke[i]}(m_j) \neq c_s$
    $\sigma_s[i] \leftarrow \mathsf{MACG}_{km[i]}(c_s[i])$
    $d[i] \leftarrow \mathsf{D}^{\mathsf{CCAKEM1}}_{\mathsf{sk}}(c[i], c_s[i], \sigma_s[i])$
    (note: we define $km'[i] = h'[i] \leftarrow H(\mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c[i]), c[i]))$
    If $d[i] \neq Rej$ then Return 1
    Else Return 0
```

We have not shown above the actions of the 'CCAExp' experiment while running A^{CC}, which are described in definition 4 . Namely, before running A^{CC}, a KEM key pair (sk, pk) is generated and $\mathsf{A}^{\mathsf{CC}}_{\mathsf{find}}$ is given the public key pk, and access to the decryption oracle $\mathsf{D}^{\mathsf{CCAKEM1}}_{\mathsf{sk}}(.)$, which makes use of the secret key sk, and the random oracle $H(.)$. When $\mathsf{A}^{\mathsf{CC}}_{\mathsf{find}}$ outputs the pair of messages m_0 and m_1, an independent and uniform bit b is chosen and the challenge ciphertext $\mathsf{E}^{\mathsf{CCAKEM1}}_{\mathsf{pk}}(m_b) = (c, c_s, \sigma_s)$ is generated. We denote by $K \overset{\text{def}}{=} \mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c)$ the encapsulated key used to generate the challenge ciphertext, and hence the portion (c_s, σ_s) of the challenge ciphertext satisfies $m_b = \mathsf{D}^{\mathsf{SYM}}_{ke}(c_s)$, and $\mathsf{MACV}_{km}(c_s, \sigma_s) = Acc$, where $km = h[1, \ldots, l_m]$ and $ke = h[l_m+1, \ldots, l_m+l_e]$ and $h = H(K, c)$. The challenge (c, c_s, σ_s) is given to $\mathsf{A}^{\mathsf{CC}}_{\mathsf{guess}}$ and when $\mathsf{A}^{\mathsf{CC}}_{\mathsf{guess}}$ outputs the guess bit b', it is compared with b and 'CCAExp' returns 1 if and only if $b' = b$.

Now we show that A^{CC} satisfies the claims of the theorem. In the following, we use the notation $\Pr[\mathsf{Event}]_{exp}$ to denote the probability of event Event in experiment exp (if no subscript is given it refers to experiment sim defined below). We first define two experiments: (1) Experiment $real$ denotes the 'OWPCAExp' experiment in definition 3 running with attacker A^{PC} whose queries are answered by the real PC Oracle. (2) Experiment sim denotes the above 'CCAExp' experiment running with attacker A^{CC} which runs A^{PC} and answers its queries with the simulator PCOSim. We define in this experiment the event $\mathsf{SuccSim}$ that the 'CCAExp' experiment returns 1. Hence $\mathbf{Succ}^{\mathrm{CCA}}_{\mathsf{A}^{\mathsf{CC}},\mathsf{CCAKEM1}}(k) \overset{\text{def}}{=} 2(\Pr[\mathsf{SuccSim}]_{sim} - \frac{1}{2})$.

Also we let $(K[j], c[j])$ denote the j'th query of A^{PC} to its PCO oracle, and we define $(K[0], c[0]) \overset{\text{def}}{=} (K, c)$ to be the challenge key-ciphertext pair.

Define the following events:

1 SuccA: $\mathsf{A}^{\mathsf{PC}}(pk, c|\mathsf{PCO}_{\mathsf{sk}}(.,.)) = \mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c)$. Note that this event is defined over the inputs to A^{PC} in both experiments *real* and *sim*.
2 Lie: $\mathsf{PCOSim}(K[j], c[j]) \neq \mathsf{PCO}_{\mathsf{sk}}(K[j], c[j])$ for some $j \in \{1, ..., q_{PC}\}$. This event is defined in experiment *sim* only.
3 Bad: $K' \overset{\text{def}}{=} \mathsf{A}^{\mathsf{PC}}(pk, c|\mathsf{PCOSim}_{\mathsf{sk}}(.,.)) \neq \mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c)$ and $d' = Acc$. This event is defined in experiment *sim* only.

We also define the event $\mathsf{Err} \overset{\text{def}}{=} \mathsf{Lie} \vee \mathsf{Bad}$. The three disjoint events Err, $\mathsf{SuccA} \wedge \neg\mathsf{Err}$ and $\neg\mathsf{SuccA} \wedge \neg\mathsf{Err}$ partition the outcome space of the *sim* experiment. Splitting event $\mathsf{SuccSim}$ we have $\Pr[\mathsf{SuccSim}] = \Pr[\mathsf{SuccSim}|\mathsf{Err}]\Pr[\mathsf{Err}] + \Pr[\mathsf{SuccSim}|\mathsf{SuccA}\wedge\neg\mathsf{Err}]\Pr[\mathsf{SuccA}\wedge\neg\mathsf{Err}] + \Pr[\mathsf{SuccSim}|\neg\mathsf{SuccA}\wedge\neg\mathsf{Err}]\Pr[\neg\mathsf{SuccA}\wedge\neg\mathsf{Err}]$. We lower bound this probability using the last two terms. If event $\mathsf{SuccA} \wedge \neg\mathsf{Err}$ occurs then A^{PC} succeeds to decrypt c, so $K' = \mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c) = K$ so $b' = b$ and $\mathsf{SuccSim}$ occurs. Hence $\Pr[\mathsf{SuccSim}|\mathsf{SuccA} \wedge \neg\mathsf{Err}] = 1$. Also, since the inputs to A^{PC} are distributed in *sim* as in *real*, we have $\Pr[\mathsf{SuccA}] = \Pr[\mathsf{SuccA}]_{real} = \mathbf{Succ}^{\mathrm{OW-PCA}}_{A^{PC},\mathsf{KEM}}(k)$ so $\Pr[\mathsf{SuccA} \wedge \neg\mathsf{Err}] = \mathbf{Succ}^{\mathrm{OW-PCA}}_{A^{PC},\mathsf{KEM}}(k) - \Pr[\mathsf{SuccA} \wedge \mathsf{Err}]$. If event $\neg\mathsf{SuccA} \wedge \neg\mathsf{Err}$ occurs then $K' \neq K$ but $d' = Rej$ so b' is chosen uniformly in $\{0, 1\}$ and hence $\Pr[\mathsf{SuccSim}|\neg\mathsf{SuccA} \wedge \neg\mathsf{Err}] = \frac{1}{2}$. Also $\Pr[\neg\mathsf{SuccA} \wedge \neg\mathsf{Err}] = \Pr[\neg\mathsf{SuccA}] - \Pr[\neg\mathsf{SuccA} \wedge \mathsf{Err}] = 1 - \mathbf{Succ}^{\mathrm{OW-PCA}}_{A^{PC},\mathsf{KEM}}(k) - \Pr[\neg\mathsf{SuccA} \wedge \mathsf{Err}]$. By substituting the above results in the last two terms of the splitting expression for $\Pr[\mathsf{SuccSim}]$ we get the lower bound $\Pr[\mathsf{SuccSim}] \geq \frac{1}{2} + \frac{1}{2}\mathbf{Succ}^{\mathrm{OW-PCA}}_{A^{PC},\mathsf{KEM}}(k) - \Pr[\mathsf{Err}]$, and hence:

$$\mathbf{Succ}^{\mathrm{CCA}}_{\mathsf{A}^{\mathsf{CC}},\mathsf{CCAKEM1}}(k) \geq \mathbf{Succ}^{\mathrm{OW-PCA}}_{A^{PC},\mathsf{KEM}}(k) - 2\Pr[\mathsf{Err}]. \tag{1}$$

The running time and query counts of the attacker A^{CC} can be readily verified. Therefore to establish the theorem it remains to show that $\Pr[\mathsf{Err}] \leq (q_{PC} + 2)\mathbf{InSec}^{\mathrm{MAC-UF}}_{\mathsf{MAC}}(t^{CC}, 0, q_{PC})$. Since $\Pr[\mathsf{Err}] = \Pr[\mathsf{Bad} \wedge \neg\mathsf{Lie}] + \Pr[\mathsf{Lie}]$ it suffices to show that

$$\Pr[\mathsf{Bad} \wedge \neg\mathsf{Lie}] \leq \mathbf{InSec}^{\mathrm{MAC-UF}}_{\mathsf{MAC}}(t^{CC}, 0, q_{PC}) \tag{2}$$

and

$$\Pr[\mathsf{Lie}] \leq (q_{PC} + 1) \cdot \mathbf{InSec}^{\mathrm{MAC-UF}}_{\mathsf{MAC}}(t^{CC}, 0, q_{PC}). \tag{3}$$

To get (2), note that Bad means that $\mathsf{MACV}_{km'}(c_s, \sigma_s) = Acc$, where $km' = H(K', c)$ and $K' \neq \mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c)$. Hence $(K', c) \neq (K, c)$ and the pair (K', c) has not been previously queried to $H(.)$ by $\mathsf{D}^{\mathsf{CCAKEM1}}_{\mathsf{sk}}$ (since all such queries have the form $(\mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(\bar{c}), \bar{c})$). Furthermore, $\mathsf{Bad}\neg\mathsf{Lie}$ means that also $(K', c) \neq (K[j], c[j])$ for all $j \in \{1, \ldots, q_{PC}\}$ since we can assume without loss of generality that A^{PC} never outputs a queried key $K[j]$ for which the query $(K[j], c)$ was answered '0' by the PCO oracle (this key is never correct). Hence when $\mathsf{Bad} \wedge \neg\mathsf{Lie}$ occurs the pair (K', c) has not been queried to $H(.)$ before $km' = H(K', c)$ is computed, so $\Pr[\mathsf{Bad} \wedge \neg\mathsf{Lie}]$ is at most the probability that $\mathsf{MACV}_{\mathsf{k}}(\sigma_s, c_s) = Acc$ when the key k is chosen uniformly in the MAC key space $\{0,1\}^{l_m}$. This probability is at most $\mathbf{InSec}_{\mathsf{MAC}}^{\mathrm{MAC-UF}}(t^{CC},0,0)$ for any choice of (c_s,σ_s). Hence $\Pr[\mathsf{Bad}\wedge\neg\mathsf{Lie}]\leq \mathbf{InSec}_{\mathsf{MAC}}^{\mathrm{MAC-UF}}(t^{CC},0,0)\leq \mathbf{InSec}_{\mathsf{MAC}}^{\mathrm{MAC-UF}}(t^{CC},0,q_{PC})$, which gives (2).

To get the second bound (3) we construct a MAC forging algorithm F as follows.

MAC Forging Attacker F Against MAC Scheme **MAC**

```
$\mathsf{F}(.|\mathsf{MACG}_{\mathsf{km}^*}(.),\mathsf{MACV}_{\mathsf{km}^*}(.))$
    $i^* \overset{\mathrm{R}}{\leftarrow} \{0,...,q_{PC}+1\}$
    $i \leftarrow 0$
    $(sk,pk) \leftarrow \mathsf{GK}(k,cp)$
    $(m_0,m_1,s) \leftarrow \mathsf{A}_{\mathsf{find}}(pk|\mathsf{D}_{\mathsf{sk}}^{\mathsf{CCAKEM1}}(.))$
    $b \overset{\mathrm{R}}{\leftarrow} \{0,1\}$ ; $r \overset{\mathrm{R}}{\leftarrow} SP_R$; $(K,c) \leftarrow \mathsf{E}_{\mathsf{pk}}^{\mathsf{KEM}}(r)$
    $h[l_m+1,\ldots,l_m+l_e] \leftarrow H(K,c)[l_m+1,\ldots,l_m+l_e]$
    $ke \leftarrow h[l_m+1,\ldots,l_m+l_e]$
    (note: if $i^*=0$ define $km=km^*$, else $km=H(K,c)[1,\ldots,l_m]$).
    $c_s \leftarrow \mathsf{E}_{ke}^{\mathsf{SYM}}(m)$
    $K' \leftarrow \mathsf{A}^{\mathsf{PC}}(pk,c|\mathsf{PCOSim}(.,.))$
    $km \leftarrow H(K,c)[1,\ldots,l_m]$; $\sigma_s \leftarrow \mathsf{MACG}_{km}(c_s)$
    (note: define $km'=km^*$)
    Return forgery $(c_s,\sigma_s)$

$\mathsf{PCOSim}(K[i],c[i]|H(.),\mathsf{D}_{\mathsf{sk}}^{\mathsf{CCAKEM1}}(.))$ (PCO Simulator for forger F)
    $i \leftarrow i+1$
    $h[i] \leftarrow H(K[i],c[i])$
    $km[i] \leftarrow h[i][1,...,l_m]$; $ke[i] \leftarrow h[l_m+1,...,l_m+l_e]$
    Find $j \in \{0,1\}$ such that $c_s[i] \overset{\mathrm{def}}{=} \mathsf{E}_{\mathsf{ke[i]}}^{\mathsf{SYM}}(m_j) \neq c_s$
    $\sigma_s[i] \leftarrow \mathsf{MACG}_{\mathsf{km[i]}}(c_s[i])$
    If $i \geq i^*$ and $c[i]=c[i^*]$ then
        $K'[i] \leftarrow \mathsf{D}_{\mathsf{sk}}^{\mathsf{KEM}}(c[i])$
        $ke'[i] \leftarrow H(K'[i],c[i])[l_m+1,\ldots,l_m+l_e]$
        (Note: we define $km'[i]=km^*$)
        $d[i] \leftarrow \mathsf{MACV}_{\mathsf{km}^*}(c_s[i],\sigma_s[i])$
        If $d[i]=Acc$ then Terminate and Return forgery $(c_s[i],\sigma_s[i])$
    Else
        $d[i] \leftarrow \mathsf{D}_{\mathsf{sk}}^{\mathsf{CCAKEM1}}(c[i],c_s[i],\sigma_s[i])$
    If $d[i] \neq Rej$ then Return 1
    Else Return 0
```

We denote by *fsim* the experiment of running the forger F in the 'MACUF-Exp' MAC forging experiment defined in the Appendix and we let SuccF denote the event that the experiment returns 1, that is F succeeds in its MAC forgery realtive to the MAC key km^* . First observe that in *sim* the PCO simulator PCOSim never lies on queries $(K[j],c[j])$ for which $\mathsf{PCO}_{\mathsf{sk}}(K[j],c[j])=1$. This is because $\mathsf{PCO}_{\mathsf{sk}}(K[j],c[j])=1$ means $K[j]=\mathsf{D}_{\mathsf{sk}}^{\mathsf{KEM}}(c[j])$ and hence the ciphertext $(c[j],c_s[j],\sigma_s[j])$ computed by PCOSim is valid and not rejected by $\mathsf{D}_{\mathsf{sk}}^{\mathsf{CCAKEM1}}$ so $\mathsf{PCOSim}(K[j],c[j])=1$. Hence Lie means $\mathsf{PCO}_{\mathsf{sk}}(K[j],c[j])=0$ but $\mathsf{PCOSim}(K[j],c[j])=1$ for some j, or equivalently $K[j]\neq \mathsf{D}_{\mathsf{sk}}^{\mathsf{KEM}}(c[j])$ and

$\mathsf{MACV}_{\mathsf{km'[j]}}(c_s[j], \sigma_s[j]) = Acc$, where $km'[j] = H(\mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c[j]), c[j])[1,\ldots,l_m]$ is the MAC key used by $\mathsf{D}^{\mathsf{CCAKEM1}}_{\mathsf{sk}}$ to check the ciphertext $(c[j], c_s[j], \sigma_s[j])$.

We now split Lie into a union of disjoint events $\mathsf{Lie}_{j,\ell}$, where $\mathsf{Lie}_{j,\ell}$ is then event that a lie first occurred at PC query j and $\ell \leq j$ is the smallest index such that $c[\ell] = c[j]$. Note that $\mathsf{Lie}_{j,\ell}$ means that (L.1) $(K[k], c[k]) \neq (\mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c[j]), c[j])$ for all $k \leq j$ (otherwise A^{PC} already knows the decryption of $c[j]$ before query j) and hence $(\mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c[j]), c[j]) = (\mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c[\ell]), c[\ell])$ was first queried to $H(.)$ by $\mathsf{D}^{\mathsf{CCAKEM1}}_{\mathsf{sk}}$ when decrypting $(c[\ell], c_s[\ell], \sigma_s[\ell])$ and (L.2) $km'[j] = km'[\ell]$ and $\mathsf{MACV}_{\mathsf{km'[j]}}(c_s[j], \sigma_s[j]) = Acc$. Therefore if we set $i^* = \ell$ in experiment *fsim*, then due to (L.1), for all outcomes in $\mathsf{Lie}_{j,\ell}$ *fsim* will run with random MAC key $km'[j] = km'[\ell] = km^*$ in the same way as *sim* runs with $km'[j] = km'[\ell] = H(\mathsf{D}^{\mathsf{KEM}}_{\mathsf{sk}}(c[\ell]), c[\ell])[1,\ldots,l_m]$. This means, for all ℓ, j that

$$\Pr[\mathsf{Lie}_{j,\ell} | i^* = \ell]_{fsim} = \Pr[\mathsf{Lie}_{j,\ell}]_{sim}, \tag{4}$$

and from (L.2) the event $\mathsf{Lie}_{j,\ell} \wedge i^* = \ell$ means that $km'[j] = km'[\ell] = km'[i^*] = km^*$ so $\mathsf{MACV}_{\mathsf{km^*}}(c_s[j], \sigma_s[j]) = Acc$ and SuccF occurs with no MAC generation queries and up to q_{PC} verify queries. So the following also holds (over all $j \in \{1,\ldots,q_{PC}\}$ and $\ell \in \{0,\ldots,j\}$):

$$\Pr[\mathsf{SuccF}]_{fsim} \geq \sum_{j,\ell} \Pr[\mathsf{Lie}_{j,\ell} \wedge i^* = \ell]_{fsim}. \tag{5}$$

Now, $\Pr[\mathsf{Lie}_{j,\ell} \wedge i^* = \ell]_{fsim} = \Pr[\mathsf{Lie}_{j,\ell} | i^* = \ell]_{fsim} \Pr[i^* = \ell]_{fsim}$ so $\Pr[\mathsf{Lie}_{j,\ell} \wedge i^* = \ell]_{fsim} = \frac{1}{q_{PC}+1} \Pr[\mathsf{Lie}_{j,\ell}]_{sim}$ for each j, ℓ using (4) and that i^* is uniformly chosen in $\{0,\ldots,q_{PC}\}$. Plugging this in (5) we get $\Pr[\mathsf{SuccF}]_{fsim} \geq \frac{1}{q_{PC}+1} \Pr[\mathsf{Lie}]_{sim}$ using $\Pr[\mathsf{Lie}]_{fsim} = \sum_{j,\ell} \Pr[\mathsf{Lie}_{j,\ell}]_{fsim}$. But on the other hand $\Pr[\mathsf{SuccF}]_{fsim} \leq \mathbf{InSec}^{\mathrm{MAC-UF}}_{\mathsf{MAC}}(t^{CC}, 0, q_{PC})$. Combining these upper and lower bounds on $\Pr[\mathsf{SuccF}]_{fsim}$ we immediately obtain the desired result (3), which completes the proof. □

As a special case of this result, when KEM is the Diffie-Hellman one in a group (see previous section), we conclude that the 'Strong Diffie-Hellman' (SDH) assumption is necessary (and sufficient, as shown in [4]) for the CCA security of the DHIES scheme.

4.2 Attack on CCAKEM2 if the underlying KEM is not OW-PCA

Using a chosen-ciphertext attack analogous to the one used to prove Theorem 1, we obtain the following result, whose proof is omitted due to lack of space.

Theorem 2. *Let A^{PC} be an attack algorithm with resource parameters (t^{PC}, q_{PC}) for breaking OW-PCA of KEM. Then we can construct an attack algorithm $\mathsf{A}^{\mathsf{CC}} = (\mathsf{A}^{\mathsf{CC}}_{\mathsf{find}}, \mathsf{A}^{\mathsf{CC}}_{\mathsf{guess}})$ with resource parameters $(t^{CC}, q_D, q_{H_1}, q_{H_2})$ such that*

$$\mathbf{Succ}^{\mathrm{CCA}}_{\mathsf{A^{CC}},\mathsf{CCAKEM2}}(k) \geq \mathbf{Succ}^{\mathrm{OW-PCA}}_{\mathsf{A},\mathsf{KEM}}(k) - \frac{2(q_{PC}+1)}{2^{l_\sigma}}$$

and $t^{CC} = t^{PC} + (q_{PC}+1)O(1)$, $q_D = q_{PC}$, $q_{H_1} = 2q_{PC}+1$, $q_{H_2} = 2q_{PC}+1$.

As a special case, when the KEM is the Diffie-Hellman one, the SDH assumption is necessary for the CCA security of REACT. As pointed out earlier, when the KEM is built from a deterministic encryption scheme, the OW-PCA assumption on the KEM is equivalent to the one-wayness assumption on the KEM so this result does not imply the necessity of stronger assumptions than one-wayness in this case (e.g. in the case of RSA).

5 Relations Between ODH and SDH Assumptions

In this section, we clarify the relation between Strong Diffie-Hellman (SDH) and Oracle Diffie-Hellman (ODH) assumptions under the random oracle model. Formal definitions for SDH and ODH are given in the appendix (definitions 6 and 7). Note that the reduction from SDH to ODH, namely, breaking SDH using ODH attacker was already shown in [4].

However, the attack on CCAKEM1 presented in the previous section implies that there exists an opposite way of reduction, i.e., a reduction from ODH to SDH: Since OW-PCA for the Diffie-Hellman KEM is exactly the same as the SDH assumption, the theorem 2 implies that there exists a reduction from CCA security for CCAKEM1 to SDH. But, in [4], the reduction from ODH to CCA security was shown in the standard model (and hence in the random oracle model) and therefore there exists a reduction from ODH to SDH. Consequently, ODH and SDH are equivalent in the random oracle model.

Apart from the trivial deduction described above we provide an explicit and tight reduction from ODH to SDH in the random oracle model.

Theorem 3. *Let $\mathsf{A}^{\mathsf{SDH}}$ be an attack algorithm with resource parameters $(t^{SDH}, q_{\mathcal{O}_x})$ for breaking SDH. Then we can construct an attack algorithm $\mathsf{A}^{\mathsf{ODH}}$ for ODH with resource parameter $(t^{ODH}, q_{\mathcal{H}_x})$ such that*

$$\mathbf{Succ}^{ODH}_{\mathsf{A}^{\mathsf{ODH}}}(k) \geq \mathbf{Succ}^{SDH}_{\mathsf{A}^{\mathsf{SDH}}}(k) - \frac{q_{\mathcal{O}_x}+1}{2^{l_h}}$$

and $t^{ODH} = t^{SDH} + (q_{\mathcal{O}_x}+1)O(1)$ and $q_{\mathcal{H}_x} = q_{\mathcal{O}_x}$. Here, l_h denotes the length of the outputs of a random oracle H.

Proof. Let $H : \{0,1\}^* \rightarrow \{0,1\}^{l_h}$ be a random oracle. Let G be a multiplicatively-written group as defined in the definitions 6 and 7. We construct an attack algorithm $\mathsf{A}^{\mathsf{ODH}}$ for breaking ODH using an attack algorithm $\mathsf{A}^{\mathsf{SDH}}$ for SDH. Note that $\mathsf{A}^{\mathsf{SDH}}$ can simulate the restricted DDH oracle $\mathcal{O}_x(.,.)$ using its oracles $H(.)$ and $\mathcal{H}_x(.)$. A complete specification for $\mathsf{A}^{\mathsf{ODH}}$ is as follows.

Attacker $\mathsf{A}^{\mathsf{ODH}}$ Against ODH Assumption

```
$\mathsf{A}^{\mathsf{ODH}}(g^r, g^x, \gamma | H(.), \mathcal{H}_x(.))$
    Run $\mathsf{A}^{\mathsf{SDH}}(g^r, g^x | \mathcal{O}_x\text{-}\mathsf{Sim}(.,.))$
    $K \leftarrow \mathsf{A}^{\mathsf{SDH}}(g^r, g^x | \mathcal{O}_x\text{-}\mathsf{Sim}(.,.))$
    If $\gamma = H(K)$ Return 1 Else Return 0

$\mathcal{O}_x\text{-}\mathsf{Sim}(c[i], w[i] | H(.), \mathcal{H}_x(.))$
    If $c[i] \neq g^r$ and $H(w[i]) = \mathcal{H}_x(c[i])$
        Return 1 Else Return 0
    If $c[i] = g^r$ and $H(g^x w[i]) = \mathcal{H}_x(gc[i])$
        Return 1 Else Return 0
```

Now we show that $\mathsf{A}^{\mathsf{ODH}}$ satisfies the claim of the theorem. We use the notation $\Pr[\mathsf{Event}]_{exp}$ to denote the probability of event Event in experiment *exp*. Let *real* denote the 'SDHExp' experiment in definition 6 running with attacker $\mathsf{A}^{\mathsf{SDH}}$ whose queries are answered by the real oracle $\mathcal{O}_x(.,.)$. We define in this experiment the event $\mathsf{SuccReal}$ that the experiment returns 1. Hence $\Pr[\mathsf{SuccReal}]_{real} \overset{\text{def}}{=} \mathbf{Succ}^{SDH}_{\mathsf{A}^{\mathsf{SDH}}}(k)$. Let *sim* denote the 'ODHRealExp' and 'ODHRandExp' experiments in definition 7 running with attacker $\mathsf{A}^{\mathsf{ODH}}$ which runs $\mathsf{A}^{\mathsf{SDH}}$ and answers its queries with the simulator $\mathcal{O}_x$-Sim. Then by definition we have $\mathbf{Succ}^{ODH}_{\mathsf{A}^{\mathsf{ODH}}}(k) = \Pr[\mathbf{ODHExpReal}(k, \mathsf{A}^{\mathsf{ODH}}) = 1]_{sim} - \Pr[\mathbf{ODHExpRrand}(k, \mathsf{A}^{\mathsf{ODH}}) = 1]_{sim}$.

First, we lower bound $\Pr[\mathbf{ODHExpReal}(k, \mathsf{A}^{\mathsf{ODH}}) = 1]_{sim}$. Now we define the following events.

– Lie: $\mathcal{O}_x$-$\mathsf{Sim}(c[j], w[j]) \neq \mathcal{O}_x(c[j], w[j])$ for some $j \in [1, ..., q_{\mathcal{O}_x}]$.

Note that if Lie does not happen in experiment *sim* $\mathsf{A}^{\mathsf{SDH}}$ cannot distinguish its environment in *real* from *sim*. Hence we get $\Pr[\mathbf{ODHExpReal}(k, \mathsf{A}^{\mathsf{ODH}}) = 1]_{sim} \geq \Pr[\mathsf{SuccReal}|\neg\mathsf{Lie}]_{sim} = \Pr[\mathsf{SuccReal}|\neg\mathsf{Lie}]_{real} \geq \Pr[\mathsf{SuccReal} \wedge \neg\mathsf{Lie}]_{real} = \Pr[\mathsf{SuccReal}]_{real} - \Pr[\mathsf{Lie}]_{real} = \Pr[\mathsf{SuccReal}]_{real} - \Pr[\mathsf{Lie}]$.

Now we upper bound $\Pr[\mathsf{Lie}]$. Assume that Lie is true: We have the following two cases (events).

– Case (1): $\mathcal{O}_x$-$\mathsf{Sim}(c[j], w[j]) = 1$ and $\mathcal{O}_x(c[j], w[j]) = 0$
– Case (2): $\mathcal{O}_x$-$\mathsf{Sim}(c[j], w[j]) = 0$ and $\mathcal{O}_x(c[j], w[j]) = 1$

From case (1), we have $H(w[j]) = \mathcal{H}_x(c[j])(= H(c[j]^x))$ but $w[j] \neq c[j]^x$ when $c[j] \neq g^r$ by the definition of $\mathcal{O}_x$-$\mathsf{Sim}(.,.)$. When $c[j] = g^r$, we have $H(g^x w[j]) = \mathcal{H}_x(gc[j])(= H(g^x c[j]^x))$. In both cases (whether $c[j] = g^r$ or not), we have $\Pr[\text{Case (1)}] = \frac{1}{2^{l_h}}$ since $H(.)$ is assumed to be a random oracle. However $\Pr[\text{Case (2)}] = 0$ as long as H is a well-defined function. Therefore we get $\Pr[\mathsf{Lie}] = \frac{q_{\mathcal{O}_x}}{2^{l_h}}$. Then we obtain $\Pr[\mathbf{ODHExpReal}(k, \mathsf{A}^{\mathsf{ODH}}) = 1]_{sim} \geq \Pr[\mathsf{SuccReal}]_{real} - \Pr[\mathsf{Lie}] = \mathbf{Succ}^{SDH}_{\mathsf{A}^{\mathsf{SDH}}}(k) - \frac{q_{\mathcal{O}_x}}{2^{l_h}}$.

Now we upper bound $\Pr[\mathbf{ODHExpRand}(k, \mathsf{A}^{\mathsf{ODH}}) = 1]$ where γ is given to $\mathsf{A}^{\mathsf{ODH}}$ as a random string of the length l_h. Since γ is uniform and independent of g^r and g^x, we have $\Pr[\mathbf{ODHExpRand}(k, \mathsf{A}^{\mathsf{ODH}}) = 1] \leq \frac{1}{2^{l_h}}$.

Then subtracting the bounds on $\Pr[\mathbf{ODHExpReal}(k, \mathsf{A}^{\mathsf{ODH}}) = 1]_{sim}$ and $\Pr[\mathbf{ODHExpRand}(k, \mathsf{A}^{\mathsf{ODH}}) = 1]$, we obtain $\mathbf{Succ}^{ODH}_{\mathsf{A}^{\mathsf{ODH}}}(k) \geq \mathbf{Succ}^{SDH}_{\mathsf{A}^{\mathsf{SDH}}}(k) - \frac{q_{\mathcal{O}_x}}{2^{l_h}} - \frac{1}{2^{l_h}}$.

As a result we obtain the following:

$$\mathbf{Succ}^{ODH}_{\mathsf{A}^{\mathsf{ODH}}}(k) \geq \mathbf{Succ}^{SDH}_{\mathsf{A}^{\mathsf{SDH}}}(k) - \frac{q_{\mathcal{O}_x} + 1}{2^{l_h}}. \tag{6}$$

The running time and query counts can be readily checked. □

We remark that the reduction from ODH to SDH still holds even if the random oracle H is replaced by a collision-resistant hash function, i.e., in the standard model. However, we were not able to find a reduction from SDH to ODH in the standard model, which implies that ODH (in the standard model) on which the CCA security of DHIES is based is a very *strong* assumption.

6 Conclusion

In this paper we clarified the necessary assumptions for the security of recently proposed schemes DHIES and REACT, and indeed for a wider class of natural asymmetric encryption schemes which include the latter two as special cases. We also clarified the relationship between ODH and SDH, two new Diffie-Hellman related assumptions. The results in this paper can be served as criteria for distinguishing the asymmetric encryption schemes whose CCA security is based on stronger assumptions such as GDH from the schemes based on weaker ones such as Computational Diffie-Hellman and Decisional Diffie-Hellman.

A Appendix

In this appendix we review the definitions of standard primitives and their security notions. These definitions are referred to in the body of the paper.

A.1 Asymmetric Encryption Schemes

Here we review the standard indistnguishability-based notion 'CCA' for the chosen ciphertext attack security for asymmetric encryption schemes, sometimes known as IND-CCA2 (see, eg. [1]).

An asymmetric encryption scheme consists of 3 algorithms: (1) A key-pair generation algorithm $\mathsf{GK}(k)$ which generates a secret/public key pair (sk, pk); (2) A probabilistic encryption algorithm $\mathsf{E}^{\mathsf{ASYM}}_{\mathsf{pk}}(m)$, which takes a public key pk and a message m and returns a ciphertext c; (3) A decryption algorithm $\mathsf{D}^{\mathsf{ASYM}}_{\mathsf{sk}}(c)$, which takes a secret key and a ciphertext c and returns a message m.

The CCA security notion is then quantitatively defined as follows.

Definition 4. (CCA) *Let* $\mathsf{ASYM} = (\mathsf{GK}, \mathsf{E}^{\mathsf{ASYM}}, \mathsf{D}^{\mathsf{ASYM}})$ *be an asymmetric encryption scheme. Let* $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ *be an attack algorithm, consisting of two 'sub-attack' algorithms* $\mathsf{A}_{\mathsf{find}}$ *and* $\mathsf{A}_{\mathsf{guess}}$. *Define the experiment*

```
Experiment $\mathbf{CCAExp}(k, \mathsf{ASYM}, \mathsf{A})$
    $(sk, pk) \leftarrow \mathsf{GK}(k)$
    $(m_0, m_1, s) \leftarrow \mathsf{A}_{\mathsf{find}}(pk|\mathsf{D}^{\mathsf{ASYM}}_{\mathsf{sk}}(.))$
    $b \overset{R}{\leftarrow} \{0,1\}$; $c \leftarrow \mathsf{E}^{\mathsf{ASYM}}_{\mathsf{pk}}(m_b)$
    $b' \leftarrow \mathsf{A}_{\mathsf{guess}}(pk, s, c|\mathsf{D}^{\mathsf{ASYM}}_{\mathsf{sk}}(.))$
    If $b' = b$ and $\mathsf{A}_{\mathsf{guess}}$ did not query $c$ to $\mathsf{D}^{\mathsf{ASYM}}_{\mathsf{sk}}(.)$ then
    Return 1 else Return 0
```

We quantify A*'s success in breaking the* CCA *security notion of scheme* ASYM *by the advantage* $\mathbf{Succ}^{\mathrm{CCA}}_{\mathsf{A},\mathsf{ASYM}}(k) \overset{\mathrm{def}}{=} 2(\Pr[\mathbf{CCAExp}(k, \mathsf{ASYM}, \mathsf{A}) = 1] - \frac{1}{2})$. *We define* A*'s resource parameters as* $RP = (t, q_D, q_{RO_1}, ..., q_{RO_n})$ *if* A *has running time/program size at most* t *and makes at most* q_D *queries to the decryption oracle, and, if the scheme makes use of* n *random oracles, at most* q_{RO_i} *queries to the* i*'th random oracle* RO_i, *for* $i \in \{1, ..., n\}$.

A.2 Message Authentication Code (MAC) Schemes

We review the definition of a MAC and its unforgeability security notion '$\mathrm{MAC}-\mathrm{UF}$'.

A MAC scheme consists of 2 algorithms: (1) A MAC generation algorithm $\mathsf{MACG}_{\mathsf{sk}}(m)$, which takes a secret key $sk \in SP_K$ and a message m and returns a MAC tag σ; (2) A MAC verification algorithm $\mathsf{MACV}_{\mathsf{sk}}(m, \sigma)$, which takes a secret key $sk \in SP_K$, a message m, and a MAC tag σ and returns a verification decision $d \in \{Acc, Rej\}$.

The $\mathrm{MAC}-\mathrm{UF}$ unforgeability security notion for a MAC scheme is then quantitatively defined as follows.

Definition 5. (MAC-UF) *Let* $\mathsf{MAC} = (\mathsf{MACG}, \mathsf{MACV})$ *be a MAC scheme with key space* SP_K. *Let* A *be an attack algorithm. Define the experiment*

```
Experiment $\mathbf{MACUFExp}(\mathsf{MAC}, \mathsf{A})$
    $sk \overset{\mathrm{R}}{\leftarrow} SP_K$
    $(m^*, \sigma^*) \leftarrow \mathsf{A}(|\mathsf{MACG}_{\mathsf{sk}}(.), \mathsf{MACV}_{\mathsf{sk}}(.))$
    If $\mathsf{MACV}_{\mathsf{sk}}(m^*, \sigma^*) = Acc$ and $\mathsf{A}$ did not query
        $m^*$ to $\mathsf{MACG}_{\mathsf{sk}}(.)$ then Return 1 Else Return 0
```

We quantify A*'s success in breaking the* $\mathrm{MAC}-\mathrm{UF}$ *security notion of scheme* MAC *by the probability* $\mathbf{Succ}^{\mathrm{MAC-UF}}_{\mathsf{A},\mathsf{MAC}} \overset{\mathrm{def}}{=} \Pr[\mathbf{MACUFExp}(\mathsf{MAC}, \mathsf{A}) = 1]$. *We quantify the insecurity of scheme* MAC *in the sense of* $\mathrm{MAC}-\mathrm{UF}$ *against arbitrary attackers with resource parameters* $RP = (t, q_{MG}, q_{MV})$ *by the probability* $\mathbf{InSec}^{\mathrm{MAC-UF}}_{\mathsf{MAC}}(t, q_{MG}, q_{MV}) \overset{\mathrm{def}}{=} \max_{\mathsf{A} \in AS_{RP}} \mathbf{Succ}^{\mathrm{MAC-UF}}_{\mathsf{A},\mathsf{MAC}}$. *The attacker set* AS_{RP} *contains all attackers with resource parameters* RP*, meaning running time+program size at most* t*, and at most* q_{MG} *queries to the MAC generation oracle and* q_{MV} *queries to the MAC verification oracle.*

A.3 Symmetric Encryption Schemes

An symmetric encryption scheme consists of 2 algorithms: (1) A probabilistic encryption algorithm $\mathsf{E}^{\mathsf{SYM}}_{\mathsf{sk}}(m)$, which takes a secret key $sk \in SP_K$ and a message m and returns a ciphertext c; (2) A decryption algorithm $\mathsf{D}^{\mathsf{SYM}}_{\mathsf{sk}}(c)$, which takes a secret key $sk \in SP_K$ and a ciphertext c and returns a message m.

A.4 Oracle Diffie-Hellman and Strong Diffie-Hellman Assumptions

We review the definition of the SDH (Strong Diffie-Hellman) and ODH (Oracle Diffie-Hellman), which are computational and decisional assumptions, respectively, defined in [4].

Definition 6. (Strong Diffie-Hellman: SDH) *Let* G *be a multiplicatively-written group. Let* $\mathsf{GenGParm}(k)$ *be an algorithm that outputs common parameters* $cp = (d_G, g, q)$ *consisting of description* d_G *of a finite cyclic group* G*, a generator* $g \in G$ *and the order* q *of* G. *Let* $\mathsf{A}^{\mathsf{SDH}}$ *be an attack algorithm. Define the experiment*

```
Experiment $\mathbf{SDHExp}(k, \mathsf{A}^{\mathsf{SDH}})$
    $cp \leftarrow \mathsf{GenGParm}(k)$
    $r, x \overset{\mathrm{R}}{\leftarrow} \{1, \ldots, q\}$; $K \leftarrow g^{rx}$
    $K' \leftarrow \mathsf{A}^{\mathsf{SDH}}(g^r, g^x | \mathcal{O}_x(.,.))$
    If $K' = K$ then Return 1 Else Return 0
```

Here, $\mathcal{O}_x(.,.)$ is a restricted DDH oracle. On input (c, w) it outputs 1 if $w = c^x$, otherwise, outputs 0. We quantify $\mathsf{A}^{\mathsf{SDH}}$'s success in computing the Diffie-Hellman key g^{rx} by the probability $\mathbf{Succ}^{SDH}_{\mathsf{A}^{\mathsf{SDH}}}(k) \overset{\mathrm{def}}{=} \Pr[\mathbf{SDHExp}(k, \mathsf{A}^{\mathsf{SDH}}) = 1]$. *We define A's resource parameters as $RP = (t, q_{\mathcal{O}_x})$ if A has running time/program size at most t and makes at most $q_{\mathcal{O}_x}$ queries to the restricted DDH oracle.*

Definition 7. (Oracle Diffie-Hellman: ODH) *Let G be a multiplicatively-written group. Let $\mathsf{GenGParm}(k)$ be an algorithm that outputs common parameters $cp = (d_G, g, q)$ consisting of description d_G of a finite cyclic group G, a generator $g \in G$ and the order q of G. A hash function $H : \{0,1\}^* \rightarrow \{0,1\}^{l_h}$, modelled as a random oracle. Let $\mathsf{A}^{\mathsf{ODH}}$ be an attack algorithm and $b' \in \{0,1\}$. Define two experiments*

```
Experiment $\mathbf{ODHExpReal}(k, \mathsf{A}^{\mathsf{ODH}})$
    $cp \leftarrow \mathsf{GenGParm}(k)$
    $r, x \overset{\mathrm{R}}{\leftarrow} \{1, \ldots, q\}$
    $b' \leftarrow \mathsf{A}^{\mathsf{ODH}}(g^r, g^x, H(g^{rx}) | H(.), \mathcal{H}_x(.))$
    Return $b'$

Experiment $\mathbf{ODHExpRand}(k, \mathsf{A}^{\mathsf{ODH}})$
    $cp \leftarrow \mathsf{GenGParm}(k)$
    $r, x \overset{\mathrm{R}}{\leftarrow} \{1, \ldots, q\}$; $h \leftarrow \{0,1\}^{l_h}$
    $b' \leftarrow \mathsf{A}^{\mathsf{ODH}}(g^r, g^x, h | H(.), \mathcal{H}_x(.))$
    Return $b'$
```

Here, $\mathcal{H}_x(c) \overset{\mathrm{def}}{=} H(c^x)$ and $\mathsf{A}^{\mathsf{SDH}}$ is not allowed to query g^r to $\mathcal{H}_x(.)$. We quantify $\mathsf{A}^{\mathsf{ODH}}$'s success in distinguishing the hash of Diffie-Hellman key g^{rx}, i.e, $H(g^{rx})$ from the random string h by the probability $\mathbf{Succ}^{ODH}_{\mathsf{A}^{\mathsf{ODH}}}(k) \overset{\mathrm{def}}{=} \Pr[\mathbf{ODHExpReal}(k, \mathsf{A}^{\mathsf{ODH}}) = 1] - \Pr[\mathbf{ODHExpRand}(k, \mathsf{A}^{\mathsf{ODH}}) = 1]$. *We define A's resource parameters as $RP = (t, q_H, q_{\mathcal{H}_x})$ if A has running time/program size at most t and makes at most q_H and $q_{\mathcal{H}_x}$ queries to the oracles $H(.)$ and $\mathcal{H}_x(.)$, respectively.*

References

1. M. Bellare, A. Desai, D. Pointcheval, and P.Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Advances in Cryptology - Proceedings of CRYPTO '98*, volume 1462 of *LNCS*, pages 26–45, Berlin, 1998. Springer-Verlag.
2. M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of First ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.
3. M. Bellare and P. Rogaway. Minimizing the use of random oracles in authenticated encryption schemes. In *Information and Communications Security*, volume 1334 of *LNCS*, pages 1–16, Berlin, 1997. Springer-Verlag.
4. M. Bellare M. Abdalla and P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158, Berlin, 2001. Springer-Verlag. See full paper available at `www-cse.ucsd.edu/users/mihir.`
5. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *LNCS*, pages 159–174, Berlin, 2001. Springer-Verlag.
6. Y. Zheng and J. Seberry. Immunizing public key cryptosystems against chosen ciphertext attacks. In the Special Issue on Secure Communications, IEEE Journal on Selected Areas in Communications, Vol. 11, No. 5, 1993, pages 715-724.
7. V. Shoup. A Proposal for an ISO Standard for Public Key Encryption (Cersion 1.1). *ISO/IEC JTC 1/SC 27*, 2001.