An extended abstract of this paper appears in *Advances in Cryptology – ASIACRYPT '04*, Lecture Notes in Computer Science Vol. ??, P. J. Lee ed., Springer-Verlag, 2004. This is the full version.

Towards Plaintext-Aware Public-Key Encryption without Random Oracles

Mihir Bellare* Adriana Palacio†

September 2004

Abstract

We consider the problem of defining and achieving plaintext-aware encryption without random oracles in the classical public-key model. We provide definitions for a hierarchy of notions of increasing strength: PA0, PA1 and PA2, chosen so that PA1+IND-CPA → IND-CCA1 and PA2+IND-CPA → IND-CCA2. Towards achieving the new notions of plaintext awareness, we show that a scheme due to Damgård [14], denoted DEG, and the "lite" version of the Cramer-Shoup scheme [13], denoted CS-lite, are both PA0 under the DHK0 assumption of [14], and PA1 under an extension of this assumption called DHK1. As a result, DEG is the most efficient proven IND-CCA1 scheme known.

Keywords: Encryption theory, plaintext awareness, chosen-ciphertext attacks, random-oracle model.

*Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: mihir@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/mihir. Supported in part by NSF grants CCR-0098123, ANR-0129617 and CCR-0208842, and by an IBM Faculty Partnership Development Award.

†Dept. of Computer Science & Engineering, University of California at San Diego, 9500 Gilman Drive, La Jolla, California 92093, USA. E-Mail: apalacio@cs.ucsd.edu. URL: http://www-cse.ucsd.edu/users/apalacio. Supported in part by above-mentioned grants of first author, and by an NSF Graduate Research Fellowship.

Contents

1 Introduction

The theory of encryption is concerned with defining and implementing notions of security for encryption schemes [24, 25, 19, 27, 29, 17]. One of the themes in its history is the emergence of notions of security of increasing strength that over time find applications and acceptance.

Our work pursues, from the same perspective, a notion that is stronger than any previous ones, namely plaintext awareness. Our goal is to strengthen the foundations of this notion by lifting it out of the random-oracle model where it currently resides. Towards this end, we provide definitions of a hierarchy of notions of plaintext awareness, relate them to existing notions, and implement some of them. We consider this a first step in the area, however, since important questions are left unresolved. We begin below by reviewing existing work and providing some motivation for our work.

1.1 Background

Intuitively, an encryption scheme is plaintext aware (PA) if the "only" way that an adversary can produce a valid ciphertext is to apply the encryption algorithm to the public key and a message. In other words, any adversary against a PA scheme that produces a ciphertext "knows" the corresponding plaintext.

Random-Oracle model work. The notion of PA encryption was first suggested by Bellare and Rogaway [7], with the motivation that PA+IND-CPA should imply IND-CCA2. That is, security against chosen-plaintext attack coupled with plaintext awareness should imply security against adaptive chosen-ciphertext attack. The intuition, namely, that if an adversary knows the plaintext corresponding to a ciphertext it produces, then a decryption oracle must be useless to it, goes back to [9, 10]. Bellare and Rogaway [7] provided a formalization of PA in the random oracle (RO) model. They asked that for every adversary A taking the public key and outputting a ciphertext, there exist an extractor that, given the same public key and a transcript of the interaction of A with its RO, is able to decrypt the ciphertext output by A. We will refer to this notion as PA-BR.

Subsequently, it was found that PA-BR was too weak for PA-BR+IND-CPA to imply IND-CCA2. Bellare, Desai, Pointcheval and Rogaway [4] traced the cause of this to the fact that PA-BR did not capture the ability of the adversary to obtain ciphertexts via eavesdropping on communications made to the receiver. (Such eavesdropping can put into the adversary's hands ciphertexts whose decryptions it does not know, lending it the ability to create other ciphertexts whose decryptions it does not know.) They provided an appropriately enhanced definition (still in the RO model) that we denote by PA-BDPR, and showed that PA-BDPR+IND-CPA $\rightarrow$ IND-CCA2.

Plaintext awareness is exploited, even though typically implicitly rather than explicitly, in the proofs of the IND-CCA2 security of numerous RO-model encryption schemes, e.g., [18, 30, 8].

PA and the RO model. By restricting the above-mentioned RO-model definitions to schemes and adversaries that do not query the RO, one obtains natural counterpart standard (i.e., non-RO) model definitions of PA. These standard-model definitions turn out, however, not to be achievable without sacrificing privacy, because the extractor can simply be used for decryption. This indicates that the use of the RO model in the definitions of [7, 4] is central.

Indeed, PA as per [7, 4] is "designed" for the RO model in the sense that the definition aims to capture certain properties of certain RO-model schemes, namely, the fact that possession of the transcript of the interaction of an adversary with its RO permits decryption of ciphertexts formed by this adversary. It is not clear what counterpart this intuition has in the standard model.

The lack of a standard-model definition of PA results in several gaps. One such arises when we consider that RO-model PA schemes are eventually instantiated to get standard-model schemes.

In that case, what property are these instantiated schemes even supposed to possess? There is no definition that we might even discuss as a target.

PA VIA KEY REGISTRATION. PA without ROs was first considered by Herzog, Liskov and Micali [23], who define and implement it in an extension of the usual public-key setting. In their setting, the sender (not just the receiver) has a public key, and, in a key-registration phase that precedes encryption, proves knowledge of the corresponding secret key to a key-registration authority via an interactive proof of knowledge. Encryption is a function of the public keys of both the sender and the receiver, and the PA extractor works by extracting the sender secret key using the knowledge extractor of the interactive proof of knowledge.

Their work also points to an application of plaintext-aware encryption where they claim the use of the latter is crucial in the sense that IND-CCA2-secure encryption does not suffice, namely to securely instantiate the ideal encryption functions of the Dolev-Yao model [16].

1.2 Our goals and motivation

The goal of this work is to provide definitions and constructions for plaintext-aware public-key encryption in the standard and classical setting of public-key encryption, namely the one where the receiver (but not the sender) has a public key, and anyone (not just a registered sender) can encrypt a message for the receiver as a function of the receiver's public key. In this setting there is no key-registration authority or key-registration protocol akin to [23].

Motivations include the following. As in the RO model, we would like a tool enabling the construction of public-key encryption schemes secure against chosen-ciphertext attack. We would also like to have some well-defined notion that can be viewed as a target for instantiated RO-model PA schemes. (One could then evaluate these schemes with regard to meeting the target.)

Additionally, we would like to enable the possibility of instantiating the ideal encryption functions of the Dolev-Yao model [16] without recourse to either random oracles or the key-registration model. (The last is an application where, as per [23], PA is required and IND-CCA2 does not suffice. However, see also [1].)

As we will see later, consideration of PA in the standard model brings other benefits, such as some insight, or at least an alternative perspective, on the design of existing encryption schemes secure against chosen-ciphertext attack. Let us now discuss our contributions.

1.3 Definitions

The first contribution of this paper is to provide definitions for plaintext-aware encryption in the standard model and standard public-key setting.

OVERVIEW. We provide a hierarchy consisting of three notions of increasing strength that we denote by PA0, PA1 and PA2. There are several motivations for this. One is that these will be seen (in conjunction with IND-CPA) to imply security against chosen-ciphertext attacks of different strengths. Another is that, as will become apparent, PA is difficult to achieve, and progress can be made by first achieving it in weaker forms. Finally, it is useful, pedagogically, to bring in new definitional elements incrementally.

A CLOSER LOOK. Our basic definitional framework considers a polynomial-time adversary $\boldsymbol{C}$, called a ciphertext creator, that takes input the public key and can query ciphertexts to an oracle. A polynomial-time algorithm $\boldsymbol{C}^*$ is said to be a successful extractor for $\boldsymbol{C}$ if it can provide replies to the oracle queries of $\boldsymbol{C}$ that are computationally indistinguishable from those provided by a decryption oracle.

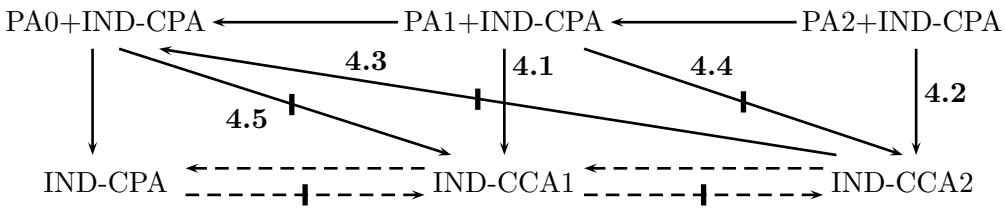

Figure 1: *An arrow is an implication, and, in the directed graph given by the arrows, there is a path from A to B if and only if A implies B. The hatched arrows represent separations. Solid lines represent results from this paper, while dashed lines represent results from prior work [4, 17]. The number on an arrow or hatched arrow refers to the theorem in this paper that establishes this relationship. Absence of a number on a solid arrow means the result is trivial.*

An important element of the above framework is that the extractor gets as input *the same public key as the ciphertext creator, as well as the coin tosses of the ciphertext creator*. This reflects the intuition that the extractor is the "subconscious" of the adversary, and begins with exactly the same information as the adversary itself.

We say that an encryption scheme is PA0 (respectively, PA1) if there exists a successful extractor for any ciphertext creator that makes only a single oracle query (respectively, a polynomial number of oracle queries).

Eavesdropping capability in PA2 is captured by providing the ciphertext creator $\boldsymbol{C}$ with an additional oracle that returns ciphertexts, but care has to be taken in defining this oracle. It does *not* suffice to let it be an encryption oracle because we want to model the ability of the adversary to obtain ciphertexts whose decryptions it may not know. Our formalization of PA2 allows the additional oracle to compute a plaintext, as a function of the query made to it and coins unknown to $\boldsymbol{C}$, and return the encryption of this plaintext to $\boldsymbol{C}$.

Formal definitions of PA0, PA1 and PA2, based on the above ideas,are in Section 3, which includes a discussion of how these definitions compare to the earlier RO-model ones.

1.4 Relations

PA by itself is not a notion of privacy, and so we are typically interested in PA coupled with the minimal notion of privacy, namely IND-CPA [24, 25]. We consider six notions, namely, PA0+IND-CPA, PA1+IND-CPA and PA2+IND-CPA, on the one hand, and the standard notions of privacy IND-CPA, IND-CCA1 [27] and IND-CCA2 [29], on the other. We provide implications and separations among these six notions in the style of [4, 17]. The results are depicted in Figure 1. For notions A, B, an implication, represented by A $\rightarrow$ B, means that every encryption scheme satisfying notion A also satisfies notion B, and a separation, represented by A $\not\rightarrow$ B, means that there exists an encryption scheme satisfying notion A but not satisfying notion B. (The latter assumes there exists some encryption scheme satisfying notion A, since otherwise the question is vacuous.)

Figure 1 shows a minimal set of arrows and hatched arrows, but the relation between any two notions is resolved by the given relations. For example, IND-CCA1 $\not\rightarrow$ PA1+IND-CPA, because, otherwise, there would be a path from IND-CCA2 to PA0+IND-CPA, contradicting the hatched arrow labeled 4.3. Similarly, we get PA0 $\not\rightarrow$ PA1 $\not\rightarrow$ PA2, meaning the three notions of plaintext awareness are of increasing strength.

The main implications are that PA1+IND-CPA implies IND-CCA1 and PA2+IND-CPA implies IND-CCA2. The PA1+IND-CPA $\rightarrow$ IND-CCA1 result shows that even a notion of PA not taking eavesdropping adversaries into account is strong enough to imply security against a significant class of chosen-ciphertext attacks. Since the PA+IND-CPA $\rightarrow$ IND-CCA2 implication has been a motivating

target for definitions of PA, the PA2+IND-CPA → IND-CCA2 result provides some validation for the definition of PA2.

Among the separations, we note that IND-CCA2 does not imply PA0, meaning even the strongest form of security against chosen-ciphertext attack is not enough to guarantee the weakest form of plaintext awareness.

1.5 Constructions

The next problem we address is to find provably-secure plaintext-aware encryption schemes.

Approaches. A natural approach to consider is to include a non-interactive zero-knowledge proof of knowledge [15] of the message in the ciphertext. However, as we explain in Section 5, this fails to achieve PA.

As such approaches are considered and discarded, it becomes apparent that achieving even the weaker forms of PA in the standard (as opposed to RO) model may be difficult. We have been able to make progress, however, under some strong assumptions that we now describe.

DHK assumptions. Let G be the order q subgroup of $\mathbb{Z}^*_{2q+1}$, where $q, 2q+1$ are primes, and let g be a generator of G. Damgård [14] introduced and used an assumption that states, roughly, that an adversary given g^a and outputting a pair of the form (g^b, g^{ab}) must "know" b. The latter is captured by requiring an extractor that given the adversary coins and inputs can output b. We call our formalization of this assumption (cf. Assumption 5.2) DHK0.[1] We also introduce an extension of this assumption called DHK1 (cf. Assumption 5.1), in which the adversary does not just output one pair (g^b, g^{ab}), but instead interacts with the extractor, feeding it such pairs adaptively and each time expecting back the discrete logarithm of the first component of the pair.

The DEG scheme. Damgård presented a simple ElGamal variant that we call DEG. It is efficient, requiring only three exponentiations to encrypt and two to decrypt.

We prove that DEG is PA0 under the DHK0 assumption and PA1 under the DHK1 assumption. Since DEG is easily seen to be IND-CPA under the DDH assumption, and we saw above that PA1+IND-CPA → IND-CCA1, a consequence is that DEG is IND-CCA1 assuming DHK1 and DDH. DEG is in fact the most efficient IND-CCA1 scheme known to date to be provably secure in the standard model.

Damgård [14] claims that DEG meets a notion of security under ciphertext attack that we call RPR-CCA1, assuming DHK0 and assuming the ElGamal scheme meets a notion called RPR-CPA. (Both notions are recalled in Appendix A, and are weaker than IND-CCA1 and IND-CPA, respectively). As we explain in Appendix A, his proof has a flaw, but his overall approach and intuition are valid, and the proof can be fixed by simply assuming DHK1 in place of DHK0. In summary, our contribution is (1) to show that DEG meets a stronger and more standard notion of security than RPR-CCA1, namely IND-CCA1, and (2) to show it is PA0 and PA1, indicating that it has even stronger properties, and providing some formal support for the intuition given in [14] about the security underlying the scheme.

CS-lite. CS-lite is a simpler and more efficient version of the Cramer-Shoup encryption scheme [13] that is IND-CCA1 under the DDH assumption. We show that CS-lite is PA0 under the DHK0 assumption

[1] Another formalization, called DA-1, is used by Hada and Tanaka [21]. (We refer to the full version of their paper [21], which points out that the formalization of the preliminary version [22] is wrong.) This differs from DHK0 in being for a non-uniform setting. DA-1 is called KEA1 by [5], based on Naor's terminology [26]: KEA stands for "knowledge of exponent." Hada and Tanaka [21] also introduced and used another assumption, that they call DA-2 and is called KEA2 in [5], but the latter show that this assumption is false. The DHK0/DA-1/KEA1 assumptions, to the best of our knowledge, are not known to be false.

and PA1 under the DHK1 assumption. (IND-CPA under DDH being easy to see, this again implies CS-lite is IND-CCA1 under DHK1 and DDH, but in this case the conclusion is not novel.) What we believe is interesting about our results is that they show that some form of plaintext awareness underlies the CS-lite scheme, and this provides perhaps an alternative viewpoint on the source of its security. We remark, however, that DEG is more efficient than CS-lite.

Warning and discussion. DHK0 and DHK1 are strong and non-standard assumptions. As pointed out by Naor [26], they are not efficiently falsifiable. (However, such assumptions can be shown to be false as exemplified in [5]). However standard-model schemes, even under strong assumptions, might provide better guarantees than RO model schemes, for we know that the latter may not provide real-world security guarantees at all [11, 28, 20, 3]. Also, PA without random oracles is challenging to achieve, and we consider it important to "break ground" by showing it is possible, even if under strong assumptions.

Open questions. The eavesdropping capability provided to an adversary in the PA2 setting seems to render the task of finding constructions significantly harder. We have not been able to find any, and leave as an open problem to find an IND-CPA+PA2 scheme provably secure under some plausible assumption. We suggest, in particular, that an interesting question is whether the Cramer-Shoup scheme, already known to be IND-CCA2, is PA2 under some appropriate assumption. (Intuitively, it seems to be PA2.) It would also be nice to achieve PA0 or PA1 under weaker and more standard assumptions than those used here.

2 Notation and standard definitions

We let $\mathbb{N} = \{1, 2, 3, \ldots\}$. We denote by ε the empty string, by $|x|$ the length of a string x, by $\bar{x}$ the bitwise complement of x, by "$\|$" the string-concatenation operator, and by 1^k the string of $k \in \mathbb{N}$ ones. We denote by $[\,]$ the empty list. Given a list L and an element x, $L \,@\, x$ denotes the list consisting of the elements in L followed by x. If S is a randomized algorithm, then $S(x, y, \ldots; R)$ denotes its output on inputs $x, y, \ldots$ and coins R; $s \xleftarrow{\$} S(x, y, \ldots)$ denotes the result of picking R at random and setting $s = S(x, y, \ldots; R)$; and $[S(x, y, \ldots)]$ denotes the set of all points having positive probability of being output by S on inputs $x, y, \ldots$. Unless otherwise indicated, an algorithm is randomized.

Encryption schemes. We recall the standard syntax. An asymmetric (also called public-key) encryption scheme is a tuple $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathrm{MsgSp})$ whose components are as follows. The polynomial-time key-generation algorithm $\mathcal{K}$ takes input 1^k, where $k \in \mathbb{N}$ is the security parameter, and returns a pair (pk, sk) consisting of a public key and matching secret key. The polynomial-time encryption algorithm $\mathcal{E}$ takes a public key pk and a message M to return a ciphertext C. The deterministic, polynomial-time decryption algorithm $\mathcal{D}$ takes a secret key sk and a ciphertext C to return either a message M or the special symbol $\perp$ indicating that the ciphertext is invalid. The polynomial-time computable message-space function MsgSp associates to each public key pk a set $\mathrm{MsgSp}(pk)$ called the message space of pk. It is required that for every $k \in \mathbb{N}$

$$\Pr\left[\, (pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)\ ;\ M \xleftarrow{\$} \mathrm{MsgSp}(pk)\ ;\ C \xleftarrow{\$} \mathcal{E}(pk, M) \ :\ \mathcal{D}(sk, C) = M \,\right] \;=\; 1\ .$$

Standard security notions. We recall the definitions of IND-CPA, IND-CCA1, and IND-CCA2 security that originate in [24], [27], and [29], respectively. We use the formalizations of [4]. Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathrm{MsgSp})$ be an asymmetric encryption scheme, let $k \in \mathbb{N}$ and $b \in \{0, 1\}$. Let $\boldsymbol{X}$ be an algorithm with access to an oracle. For $\mathrm{aaa} \in \{\mathrm{cpa}, \mathrm{cca1}, \mathrm{cca2}\}$, consider the following experiment

Experiment $\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}}(k)$

$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$; $x \xleftarrow{\$} \boldsymbol{C}^{\mathcal{D}(sk,\cdot)}(pk)$; $d \xleftarrow{\$} \boldsymbol{D}(x)$; Return d

---

Experiment $\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)$

$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$; Choose coins $R[\boldsymbol{C}], R[\boldsymbol{C}^*]$ for $\boldsymbol{C}, \boldsymbol{C}^*$, respectively ; $\mathrm{St}[\boldsymbol{C}^*] \leftarrow (pk, R[\boldsymbol{C}])$

Run $\boldsymbol{C}$ on input pk and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle queries as follows:

– If $\boldsymbol{C}$ makes query Q then

$(M, \mathrm{St}[\boldsymbol{C}^*]) \leftarrow \boldsymbol{C}^*(Q, \mathrm{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$; Return M to $\boldsymbol{C}$ as the reply EndIf

Let x denote the output of $\boldsymbol{C}$; $d \xleftarrow{\$} \boldsymbol{D}(x)$; Return d

Figure 2: Experiments used to define PA1 and PA0.

Experiment $\mathbf{Exp}^{\text{ind-aaa-}b}_{\mathcal{AE},\boldsymbol{X}}(k)$

$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$; $(M_0, M_1, \mathrm{St}) \xleftarrow{\$} \boldsymbol{X}^{\mathcal{O}_1(\cdot)}(\mathsf{find}, pk)$; $C \xleftarrow{\$} \mathcal{E}(pk, M_b)$

$d \leftarrow \boldsymbol{X}^{\mathcal{O}_2(\cdot)}(\mathsf{guess}, C, \mathrm{St})$; Return d

where

If aaa = cpa	then	$\mathcal{O}_1(\cdot) = \varepsilon$	and	$\mathcal{O}_2(\cdot) = \varepsilon$
If aaa = cca1	then	$\mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot)$	and	$\mathcal{O}_2(\cdot) = \varepsilon$
If aaa = cca2	then	$\mathcal{O}_1(\cdot) = \mathcal{D}_{sk}(\cdot)$	and	$\mathcal{O}_2(\cdot) = \mathcal{D}_{sk}(\cdot)$

In each case it is required that $M_0, M_1 \in \mathrm{MsgSp}(pk)$ and $|M_0| = |M_1|$. In the case of IND-CCA2, it is also required that $\boldsymbol{X}$ not query its decryption oracle with ciphertext C. We call $\boldsymbol{X}$ an *ind-aaa-adversary*. The *ind-aaa-advantage* of $\boldsymbol{X}$ is

$$\mathbf{Adv}^{\text{ind-aaa}}_{\mathcal{AE},\boldsymbol{X}}(k) \;=\; \Pr\left[\mathbf{Exp}^{\text{ind-aaa-1}}_{\mathcal{AE},\boldsymbol{X}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{ind-aaa-0}}_{\mathcal{AE},\boldsymbol{X}}(k) = 1\right].$$

For $\mathrm{AAA} \in \{\mathrm{CPA}, \mathrm{CCA1}, \mathrm{CCA2}\}$, $\mathcal{AE}$ is said to be IND-AAA secure if $\mathbf{Adv}^{\text{ind-aaa}}_{\mathcal{AE},\boldsymbol{X}}(\cdot)$ is negligible for every polynomial-time ind-aaa-adversary $\boldsymbol{X}$.

3 New notions of plaintext awareness

In this section we provide our formalizations of plaintext-aware encryption. We provide the formal definitions first and explanations later. We begin with PA1, then define PA0 via this, and finally define PA2.

Definition 3.1 [**PA1**] Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathrm{MsgSp})$ be an asymmetric encryption scheme. Let $\boldsymbol{C}$ be an algorithm that has access to an oracle, takes as input a public key pk, and returns a string. Let $\boldsymbol{D}$ be an algorithm that takes a string and returns a bit. Let $\boldsymbol{C}^*$ be an algorithm that takes a string and some state information, and returns a message or the symbol $\perp$, and a new state. We call $\boldsymbol{C}$ a *ciphertext-creator* adversary, $\boldsymbol{D}$ a *distinguisher*, and $\boldsymbol{C}^*$ a *pa1-extractor*. For $k \in \mathbb{N}$, we define the experiments shown in Figure 2. The *pa1-advantage* of $\boldsymbol{C}$ relative to $\boldsymbol{D}$ and $\boldsymbol{C}^*$ is

$$\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) \;=\; \Pr\left[\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}}(k) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) = 1\right].$$

We say that $\boldsymbol{C}^*$ is a *successful pa1-extractor for* $\boldsymbol{C}$ if for every polynomial-time distinguisher $\boldsymbol{D}$ the function $\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(\cdot)$ is negligible. We say $\mathcal{AE}$ is *PA1 secure* if for any polynomial-time ciphertext creator there exists a successful polynomial-time pa1-extractor. ∎

Experiment $\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P},\boldsymbol{D}}(k)$

$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$; $\textsc{List} \leftarrow [\,]$
Choose coins $R[\boldsymbol{C}], R[\boldsymbol{P}]$ for $\boldsymbol{C}, \boldsymbol{P}$, respectively ; $\text{St}[\boldsymbol{P}] \leftarrow \varepsilon$
Run $\boldsymbol{C}$ on input pk and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle queries as follows:
- If $\boldsymbol{C}$ makes query (dec, Q) then
  $M \leftarrow \mathcal{D}(sk, Q)$; Return M to $\boldsymbol{C}$ as the reply EndIf
- If $\boldsymbol{C}$ makes query (enc, Q) then
  $(M, \text{St}[\boldsymbol{P}]) \leftarrow \boldsymbol{P}(Q, \text{St}[\boldsymbol{P}]; R[\boldsymbol{P}])$; $C \xleftarrow{\$} \mathcal{E}(pk, M)$; $\textsc{List} \leftarrow \textsc{List} \mathbin{@} C$
  Return C to $\boldsymbol{C}$ as the reply EndIf

Let x denote the output of $\boldsymbol{C}$; $d \xleftarrow{\$} \boldsymbol{D}(x)$; Return d

---

Experiment $\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P},\boldsymbol{D},\boldsymbol{C}^*}(k)$

$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$; $\textsc{List} \leftarrow [\,]$
Choose coins $R[\boldsymbol{C}], R[\boldsymbol{P}], R[\boldsymbol{C}^*]$ for $\boldsymbol{C}, \boldsymbol{P}, \boldsymbol{C}^*$, respectively ; $\text{St}[\boldsymbol{P}] \leftarrow \varepsilon$; $\text{St}[\boldsymbol{C}^*] \leftarrow (pk, R[\boldsymbol{C}])$
Run $\boldsymbol{C}$ on input pk and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle queries as follows:
- If $\boldsymbol{C}$ makes query (dec, Q) then
  $(M, \text{St}[\boldsymbol{C}^*]) \leftarrow \boldsymbol{C}^*(Q, \textsc{List}, \text{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$; Return M to $\boldsymbol{C}$ as the reply EndIf
- If $\boldsymbol{C}$ makes query (enc, Q) then
  $(M, \text{St}[\boldsymbol{P}]) \leftarrow \boldsymbol{P}(Q, \text{St}[\boldsymbol{P}]; R[\boldsymbol{P}])$; $C \xleftarrow{\$} \mathcal{E}(pk, M)$; $\textsc{List} \leftarrow \textsc{List} \mathbin{@} C$
  Return C to $\boldsymbol{C}$ as the reply EndIf

Let x denote the output of $\boldsymbol{C}$; $d \xleftarrow{\$} \boldsymbol{D}(x)$; Return d

Figure 3: Experiments used to define PA2.

Definition 3.2 [**PA0**] Let $\mathcal{AE}$ be an asymmetric encryption scheme. We call a ciphertext-creator adversary that makes *exactly one* oracle query a *pa0 ciphertext creator*. We call a pa1-extractor for a pa0 ciphertext creator a *pa0-extractor*. We say that $\mathcal{AE}$ is *PA0 secure* if for any polynomial-time pa0 ciphertext creator there exists a successful polynomial-time pa0-extractor. ▌

We now explain the ideas behind the above formalisms. The core of the formalization of plaintext awareness of asymmetric encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \text{MsgSp})$ considers a polynomial-time ciphertext-creator adversary $\boldsymbol{C}$ that takes input a public key pk, has access to an oracle and returns a string. The adversary tries to distinguish between the cases that its oracle is $\mathcal{D}(sk, \cdot)$, or it is an *extractor* algorithm $\boldsymbol{C}^*$ that takes as input the same public key pk. PA1 security requires that there exist a polynomial-time $\boldsymbol{C}^*$ such that $\boldsymbol{C}$'s outputs in the two cases are indistinguishable. We allow $\boldsymbol{C}^*$ to be stateful, maintaining state $\text{St}[\boldsymbol{C}^*]$ across invocations. Importantly, $\boldsymbol{C}^*$ *is provided with the coin tosses of* $\boldsymbol{C}$; otherwise, $\boldsymbol{C}^*$ would be functionally equivalent to the decryption algorithm and thus could not exist unless $\mathcal{AE}$ were insecure with regard to providing privacy. We remark that this formulation is stronger than one not involving a distinguisher $\boldsymbol{D}$, in which $\boldsymbol{C}$ simply outputs a bit representing its guess, since $\boldsymbol{C}^*$ gets the coins of $\boldsymbol{C}$, but not the coins of $\boldsymbol{D}$.

PA0 security considers only adversaries that make a *single* query in their attempt to determine if the oracle is a decryption oracle or an extractor.

Definition 3.3 [**PA2**] Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \text{MsgSp})$ be an asymmetric encryption scheme. Let $\boldsymbol{C}$ be an algorithm that has access to an oracle, takes as input a public key pk, and returns a string. Let $\boldsymbol{P}$ be

an algorithm that takes a string and some state information, and returns a message and a new state. Let $\boldsymbol{D}$ be an algorithm that takes a string and returns a bit. Let $\boldsymbol{C}^*$ be an algorithm that takes a string, a list of strings and some state information, and returns a message or the symbol $\perp$, and a new state. We call $\boldsymbol{C}$ a *ciphertext-creator* adversary, $\boldsymbol{P}$ a *plaintext-creator* adversary, $\boldsymbol{D}$ a *distinguisher*, and $\boldsymbol{C}^*$ a *pa2-extractor*. For $k \in \mathbb{N}$, we define the experiments shown in Figure 3. It is required that, in these experiments, $\boldsymbol{C}$ not make a query (dec, C) for which $C \in \textsc{Clist}$. The *pa2-advantage* of $\boldsymbol{C}$ relative to $\boldsymbol{P}$, $\boldsymbol{D}$ and $\boldsymbol{C}^*$ is

$$\mathbf{Adv}^{\mathrm{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P},\boldsymbol{D},\boldsymbol{C}^*}(k) \;=\; \Pr\left[\mathbf{Exp}^{\mathrm{pa2\text{-}d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P},\boldsymbol{D}}(k)=1\right] - \Pr\left[\mathbf{Exp}^{\mathrm{pa2\text{-}x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P},\boldsymbol{D},\boldsymbol{C}^*}(k)=1\right].$$

We say that $\boldsymbol{C}^*$ is a *successful pa2-extractor for* $\boldsymbol{C}$ if for every polynomial-time plaintext creator $\boldsymbol{P}$ and distinguisher $\boldsymbol{D}$, the function $\mathbf{Adv}^{\mathrm{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P},\boldsymbol{D},\boldsymbol{C}^*}(\cdot)$ is negligible. We say $\mathcal{AE}$ is *PA2 secure* if for any polynomial-time ciphertext creator there exists a successful polynomial-time pa2-extractor. ▌

In the definition of PA2, the core setting of PA1 is enhanced to model the real-life capability of a ciphertext creator to obtain ciphertexts via eavesdropping on communications made by a third party to the receiver (cf. [4]). Providing $\boldsymbol{C}$ with an encryption oracle does not capture this because eavesdropping puts into $\boldsymbol{C}$'s hands ciphertexts of which it does *not* know the corresponding plaintext, and, although we disallow $\boldsymbol{C}$ to query these to its oracle, it might be able to use them to create other ciphertexts whose corresponding plaintext it does not know and on which the extractor fails.

Modeling eavesdropping requires balancing two elements: providing $\boldsymbol{C}$ with a capability to obtain ciphertexts of plaintexts it does not know, yet capturing the fact that $\boldsymbol{C}$ might have partial information about the plaintexts, or control of the distribution from which these plaintexts are drawn. We introduce a companion *plaintext-creator* adversary $\boldsymbol{P}$ who, upon receiving a communication from $\boldsymbol{C}$, creates a plaintext and forwards it to an encryption oracle. The ciphertext emanating from the encryption oracle is sent to both $\boldsymbol{C}$ and $\boldsymbol{C}^*$. $\boldsymbol{C}$ has some control over $\boldsymbol{P}$ via its communication to $\boldsymbol{P}$, but we ensure this is not total by *denying* $\boldsymbol{C}$ *and* $\boldsymbol{C}^*$ *the coin tosses of* $\boldsymbol{P}$, and also by asking that $\boldsymbol{C}^*$ depend on $\boldsymbol{C}$ but not on $\boldsymbol{P}$.

The extractor $\boldsymbol{C}^*$ is, as before, provided with the coin tosses of $\boldsymbol{C}$. Two types of oracle queries are allowed to $\boldsymbol{C}$. Via a query (dec, Q), it can ask its oracle to decrypt ciphertext Q. Alternatively, it can make a query (enc, Q) to call $\boldsymbol{P}$ with argument Q, upon which the latter computes a message M and forwards it to the encryption oracle, which returns the resulting ciphertext to $\boldsymbol{C}$, and $\boldsymbol{C}^*$ in the case that $\boldsymbol{C}$'s oracle is $\boldsymbol{C}^*$. We observe that if an asymmetric encryption scheme is PA2 secure then it is PA1 secure, and if it is PA1 secure then it is PA0 secure.

Comparison. The RO-model definitions PA-BR [7] and PA-BDPR [4] differ from ours in the following ways.

The RO-model definitions did not give the extractor the coins of the ciphertext creator. As we indicated above, in the absence of ROs, providing the extractor with the coins of the ciphertext creator is necessary for the non-triviality of PA, since otherwise the extractor can be used for decryption and the scheme will not be IND-CPA secure. Furthermore, the basic intuition is that the extractor is the subconscious of the ciphertext creator, and thus should have all the inputs of the latter, meaning it should be given the public key and the coins that were given to the ciphertext creator.

The RO-model definitions required the extractor to return the decryption of a given ciphertext. We have weakened this requirement, asking only that the outputs of the extractor be computationally indistinguishable from the outputs of the decryption oracle, because this weakening preserves the main implications and applications of PA while increasing the possibility of finding constructions. However, as discussed below, one can consider a stronger, statistical version of our definitions which captures requiring the extractor to return correct decryptions, and this might be useful in some contexts.

The RO-model definitions only consider ciphertext creators that output a single ciphertext which the extractor must decrypt, while we have considered an oracle-based formulation, corresponding to ciphertext creators that adaptively create multiple ciphertexts for the extractor to decrypt.

The most important changes are with regard to modeling eavesdropping. In PA-BDPR [4], eavesdropping capability was modeled by providing the ciphertext creator with an encryption oracle but denying it and the extractor the so-called indirect random-oracle queries, namely those made by the encryption oracle. Adopting this approach in the absence of ROs would reduce to providing the ciphertext creator with an encryption oracle, which, as we discussed above, does not correctly model eavesdropping because the ciphertext creator knows the decryptions of ciphertexts it obtains via an encryption oracle, and we want to provide it a means of obtaining ciphertexts whose decryptions it may not know. In particular, the encryption-oracle-based notion seems too weak to prove Theorem 4.2. We have used the plaintext creator instead.

The plaintext extractor in PA-BDPR [4] is black box, meaning there is a single extractor that works for all ciphertext creators, upon being given the transcript of interactions of the ciphertext creator with its oracles. We have weakened this requirement, allowing the extractor code to depend non-uniformly on the code of the ciphertext creator. Again, this was done in order to increase the possibility of finding constructions. Evidence of the power of non-black-box formulations is provided, in another context, by [2].

We note that it is easy to "lift" our standard-model definitions to counterpart RO-model definitions following the paradigm of [6]. Call these PA0-RO, PA1-RO and PA2-RO. Given the above, we suggest that these make more suitable RO model notions than the existing PA-BR [7] and PA-BDPR [4] ones, since they are rooted in the standard model rather than being RO-model definitions with no standard-model counterparts. We remark that PA-BR implies PA0-RO and PA-BDPR implies PA2-RO, which says that we have weakened the definitions. Yet we are still in line with the original intuition and have preserved the important implications and application potential.

Statistical PA. Stronger versions of our definitions of PA0, PA1 and PA2 are obtained by requiring that the outputs of the ciphertext creator, in the case where its oracle is the decryption algorithm and in the case where it is the extractor, are statistically rather than computationally indistinguishable. Formally, this can be captured by simply allowing the distinguisher to be computationally unlimited. In other words, let us say $\mathcal{AE}$ is *sPA1 secure* if for any polynomial-time ciphertext creator $\boldsymbol{C}$ there exists a polynomial-time extractor $\boldsymbol{C}^*$ such that the function $\mathbf{Adv}^{\mathrm{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(\cdot)$ is negligible for every (not necessarily polynomial time) distinguisher $\boldsymbol{D}$. We can define sPA0 and sPA2 analogously.

The statistical versions of our definitions amount simply to saying that the extractor must return the correct decryption of any ciphertext it is given, except with negligible probability. Accordingly, this could certainly have been formulated in a simpler way without introducing distinguishers at all, and, indeed, it may have been pedagogically preferable to begin with this simpler and stronger definition and only then get to our current ones. We preferred the current distinguisher-based approach because it allows us to fit the computational and statistical settings into a common definitional framework.

As indicated above, we have chosen to make the computational versions of the definitions the main ones because they suffice for the applications of Theorems 4.1 and 4.2, and might make future constructions easier to find. We remark, however, that the schemes DEG and CS-lite that we show to achieve PA0 or PA1 actually achieve sPA0 and sPA1 under the same assumptions, meaning that Theorems 5.3, and 5.4 are true if we replace PA0 by sPA0 and PA1 by sPA1.

4 Relations among notions

We now state the formal results corresponding to Figure 1, beginning with the two motivating applications of our notions of plaintext awareness. The proof of the following is in Section 4.1.

Theorem 4.1 [PA1+IND-CPA ⇒ IND-CCA1] Let $\mathcal{AE}$ be an asymmetric encryption scheme. If $\mathcal{AE}$ is PA1 secure and IND-CPA secure, then it is IND-CCA1 secure. ▌

The proof of the following is in Section 4.2.

Theorem 4.2 [PA2+IND-CPA ⇒ IND-CCA2] Let $\mathcal{AE}$ be an asymmetric encryption scheme. If $\mathcal{AE}$ is PA2 secure and IND-CPA secure, then it is IND-CCA2 secure. ▌

It is natural to ask whether the converse of Theorem 4.2 (respectively, Theorem 4.1) is true, namely whether an asymmetric encryption scheme that is IND-CCA2 (respectively, IND-CCA1) secure is also PA2 (respectively, PA1) secure. (It is, of course, IND-CPA secure). The answer is no. The following theorem implies that PA2 (respectively, PA1) security (in conjunction with IND-CPA security) is a strictly stronger requirement than IND-CCA2 (respectively, IND-CCA1) security, unless there simply do not exist any IND-CCA2-secure schemes. The proof is in Section 4.3.

Theorem 4.3 [IND-CCA2 ⇏ PA0+IND-CPA] Assume there exists an IND-CCA2-secure asymmetric encryption scheme. Then there exists an IND-CCA2-secure asymmetric encryption scheme that is *not* PA0 secure. ▌

We remind the reader that each notion of PA in the absence of IND-CPA security is trivial to achieve. (In particular, the encryption scheme in which the encryption function sets the ciphertext equal to the plaintext is PA2 secure, but not IND-CPA secure.) Thus the fact that the scheme guaranteed by Theorem 4.3 is IND-CPA secure is important.

To complete the picture of implications and separations between the PA+IND-CPA notions and the IND-AAA notions, we now show that PA1 security (in conjunction with IND-CPA security) is not sufficient to achieve IND-CCA2 security, and PA0 security (in conjunction with IND-CPA security) is not sufficient to achieve IND-CCA1 security. The proof of the following is in Section 4.4.

Theorem 4.4 [PA1+IND-CPA ⇏ IND-CCA2] Assume there exists a PA1 secure and IND-CPA-secure asymmetric encryption scheme. Then there exists a PA1 secure and IND-CPA-secure asymmetric encryption scheme that is *not* IND-CCA2 secure. ▌

The proof of the following is in Section 4.5.

Theorem 4.5 [PA0+IND-CPA ⇏ IND-CCA1] Assume there exists a PA0 secure and IND-CPA-secure asymmetric encryption scheme. Then there exists a PA0 secure and IND-CPA-secure asymmetric encryption scheme that is *not* IND-CCA1 secure. ▌

4.1 Proof of Theorem 4.1

Assume that $\mathcal{AE}$ is PA1 secure and IND-CPA secure, and let $\boldsymbol{X}$ be a polynomial-time ind-cca1-adversary attacking $\mathcal{AE}$. We construct a polynomial-time ciphertext creator $\boldsymbol{C}$ for $\mathcal{AE}$, based on $\boldsymbol{X}$, and let $\boldsymbol{C}^*$ be a successful polynomial-time pa1-extractor for it. Then we construct a polynomial-time ind-cpa-adversary $\boldsymbol{Y}$ for $\mathcal{AE}$, based on $\boldsymbol{X}$ and $\boldsymbol{C}^*$. Finally, we construct polynomial-time distinguishers $\boldsymbol{D}_0$ and $\boldsymbol{D}_1$ for $\boldsymbol{C}$, and prove that for every $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{ind-cca1}}_{\mathcal{AE},\boldsymbol{X}}(k) \;\leq\; \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE},\boldsymbol{Y}}(k) + \mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0,\boldsymbol{C}^*}(k) + \mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1,\boldsymbol{C}^*}(k)\,. \tag{1}$$

Ciphertext creator $\boldsymbol{C}(pk; R[\boldsymbol{C}])$
Run $\boldsymbol{X}(\mathsf{find}, \cdot)$ on input pk and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle queries as follows:
– If $\boldsymbol{X}(\mathsf{find}, \cdot)$ makes query Q then
Make query Q ; Upon receiving a response M, return M to $\boldsymbol{X}(\mathsf{find}, \cdot)$ as the reply EndIf
Let (M_0, M_1, St) denote the output of $\boldsymbol{X}(\mathsf{find}, \cdot)$
Return $(M_0, M_1, \mathrm{St}, pk)$

Adversary $\boldsymbol{Y}(\mathsf{find}, pk; R[\boldsymbol{Y}])$
Parse $R[\boldsymbol{Y}]$ as $R[\boldsymbol{C}] \| R[\boldsymbol{C}^*]$; $\mathrm{St}[\boldsymbol{C}^*] \leftarrow (pk, R[\boldsymbol{C}])$
Run $\boldsymbol{X}(\mathsf{find}, \cdot)$ on input pk and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle queries as follows:
– If $\boldsymbol{X}(\mathsf{find}, \cdot)$ makes query Q then
$(M, \mathrm{St}[\boldsymbol{C}^*]) \leftarrow \boldsymbol{C}^*(Q, \mathrm{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$; Return M to $\boldsymbol{X}(\mathsf{find}, \cdot)$ as the reply EndIf
Let (M_0, M_1, St) denote the output of $\boldsymbol{X}(\mathsf{find}, \cdot)$
Return (M_0, M_1, St)

Adversary $\boldsymbol{Y}(\mathsf{guess}, C, \mathrm{St})$
$d \leftarrow \boldsymbol{X}(\mathsf{guess}, C, \mathrm{St})$
Return d

Distinguisher $\boldsymbol{D}_0(x)$
Parse x as $(M_0, M_1, \mathrm{St}, pk)$
$C \xleftarrow{\$} \mathcal{E}(pk, M_0)$; $d \leftarrow \boldsymbol{X}(\mathsf{guess}, C, \mathrm{St})$
Return $\bar{d}$

Distinguisher $\boldsymbol{D}_1(x)$
Parse x as $(M_0, M_1, \mathrm{St}, pk)$
$C \xleftarrow{\$} \mathcal{E}(pk, M_1)$; $d \leftarrow \boldsymbol{X}(\mathsf{guess}, C, \mathrm{St})$
Return d

Figure 4: Ciphertext-creator adversary $\boldsymbol{C}$, ind-cpa-adversary $\boldsymbol{Y}$, and distinguishers $\boldsymbol{D}_0, \boldsymbol{D}_1$ for the proof of Theorem 4.1.

The assumption that $\mathcal{AE}$ is PA1 secure and IND-CPA secure implies that $\mathbf{Adv}^{\text{ind-cca1}}_{\mathcal{AE},\boldsymbol{X}}(\cdot)$ is negligible, and thus that $\mathcal{AE}$ is IND-CCA1 secure. We proceed to the constructions and analysis.

The four algorithms we construct are defined in Figure 4. Clearly, they all run in polynomial time. Ciphertext-creator adversary $\boldsymbol{C}$ is essentially the same as $\boldsymbol{X}(\mathsf{find}, \cdot)$, except that it returns the public key along with M_0, M_1, St. By the assumption that $\mathcal{AE}$ is PA1 secure, there is a successful polynomial-time pa1-extractor $\boldsymbol{C}^*$ for $\boldsymbol{C}$.

A random tape $R[\boldsymbol{C}] \| R[\boldsymbol{C}^*]$ for ind-cpa-adversary $\boldsymbol{Y}$ has two parts, one being a random tape for $\boldsymbol{C}$ (equivalently, for $\boldsymbol{X}$) and the other being a random tape for $\boldsymbol{C}^*$. $\boldsymbol{Y}(\mathsf{find}, \cdot)$ initializes and then maintains state for $\boldsymbol{C}^*$. It runs $\boldsymbol{X}(\mathsf{find}, \cdot)$, and if the latter makes a query, then $\boldsymbol{Y}(\mathsf{find}, \cdot)$ runs $\boldsymbol{C}^*$ to compute a reply, which it returns to $\boldsymbol{X}(\mathsf{find}, \cdot)$. When $\boldsymbol{X}(\mathsf{find}, \cdot)$ stops, $\boldsymbol{Y}(\mathsf{find}, \cdot)$ returns the former's output. $\boldsymbol{Y}(\mathsf{guess}, \cdot)$ is identical to $\boldsymbol{X}(\mathsf{guess}, \cdot)$.

Consider distinguishers $\boldsymbol{D}_0$ and $\boldsymbol{D}_1$. Intuitively, for $b \in \{0, 1\}$, when $\boldsymbol{D}_b$ is run on the output of $\boldsymbol{C}$ after the latter has interacted with the decryption oracle, $\boldsymbol{D}_0$ computes the complement of the outcome of experiment $\mathbf{Exp}^{\text{ind-cca1-0}}_{\mathcal{AE},\boldsymbol{X}}(k)$, and $\boldsymbol{D}_1$ computes the outcome of experiment $\mathbf{Exp}^{\text{ind-cca1-1}}_{\mathcal{AE},\boldsymbol{X}}(k)$.

We claim that Equation (1) holds for all $k \in \mathbb{N}$. To prove this, fix $k \in \mathbb{N}$. We state four claims, conclude the proof given them, and then return to prove the claims. The first two claims relate the probability that $\boldsymbol{X}$ guesses the value of challenge bit b, in each of its experiments, to the probability that distinguisher $\boldsymbol{D}_b$ returns 1 when it is run on the output of $\boldsymbol{C}$ after the latter has interacted with

the decryption oracle, in experiment $\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_b}(k)$.

Claim 4.6 $\Pr\left[\,\mathbf{Exp}^{\text{ind-cca1-1}}_{\mathcal{AE},\boldsymbol{X}}(k)=1\,\right] \;=\; \Pr\left[\,\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1}(k)=1\,\right]$. ∎

Claim 4.7 $\Pr\left[\,\mathbf{Exp}^{\text{ind-cca1-0}}_{\mathcal{AE},\boldsymbol{X}}(k)=1\,\right] \;=\; 1-\Pr\left[\,\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0}(k)=1\,\right]$. ∎

The other claims relate the probability that $\boldsymbol{Y}$ guesses the value of challenge bit b, in each of its experiments, to the probability that distinguisher $\boldsymbol{D}_b$ returns 1 when it is run on the output of $\boldsymbol{C}$ after the latter has interacted with pa2-extractor $\boldsymbol{C}^*$, in experiment $\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_b,\boldsymbol{C}^*}(k)$.

Claim 4.8 $\Pr\left[\,\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{AE},\boldsymbol{Y}}(k)=1\,\right] \;=\; \Pr\left[\,\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1,\boldsymbol{C}^*}(k)=1\,\right]$. ∎

Claim 4.9 $\Pr\left[\,\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{AE},\boldsymbol{Y}}(k)=1\,\right] \;=\; 1-\Pr\left[\,\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0,\boldsymbol{C}^*}(k)=1\,\right]$. ∎

Applying these claims, we obtain Equation (1) as follows:

$$\begin{aligned}
\mathbf{Adv}^{\text{ind-cca1}}_{\mathcal{AE},\boldsymbol{X}}(k) &= \Pr\left[\,\mathbf{Exp}^{\text{ind-cca1-1}}_{\mathcal{AE},\boldsymbol{X}}(k)=1\,\right]-\Pr\left[\,\mathbf{Exp}^{\text{ind-cca1-0}}_{\mathcal{AE},\boldsymbol{X}}(k)=1\,\right]\\
&= \Pr\left[\,\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1}(k)=1\,\right]-\left(1-\Pr\left[\,\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0}(k)=1\,\right]\right)\\
&= \left(\Pr\left[\,\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1,\boldsymbol{C}^*}(k)=1\,\right]+\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1,\boldsymbol{C}^*}(k)\right)-1\\
&\quad+\left(\Pr\left[\,\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0,\boldsymbol{C}^*}(k)=1\,\right]+\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0,\boldsymbol{C}^*}(k)\right)\\
&= \Pr\left[\,\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1,\boldsymbol{C}^*}(k)=1\,\right]-\left(1-\Pr\left[\,\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0,\boldsymbol{C}^*}(k)=1\,\right]\right)\\
&\quad+\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0,\boldsymbol{C}^*}(k)+\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1,\boldsymbol{C}^*}(k)\\
&= \Pr\left[\,\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{AE},\boldsymbol{Y}}(k)=1\,\right]-\Pr\left[\,\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{AE},\boldsymbol{Y}}(k)=1\,\right]\\
&\quad+\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0,\boldsymbol{C}^*}(k)+\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1,\boldsymbol{C}^*}(k)\\
&= \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE},\boldsymbol{Y}}(k)+\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0,\boldsymbol{C}^*}(k)+\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1,\boldsymbol{C}^*}(k)\;.
\end{aligned}$$

It remains to prove the four claims above.

Proof of Claim 4.6: Let $\mathcal{S}_{\text{cca1-1}}$ denote the sample space underlying $\mathbf{Exp}^{\text{ind-cca1-1}}_{\mathcal{AE},\boldsymbol{X}}(k)$. A member of this space is a string specifying coin tosses for all algorithms involved, which in this case means the coins of the key-generation algorithm, the random tape of $\boldsymbol{X}$ itself, and the coins used by the encryption algorithm.

A member of the sample space $\mathcal{S}_{\text{pa1-d-1}}$ underlying $\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1}(k)$ is a string specifying the coins of the key-generation algorithm, the random tape of $\boldsymbol{C}$, and the random tape of $\boldsymbol{D}_1$. Claim 4.6 follows once we observe that by the definitions of $\boldsymbol{C}$ and $\boldsymbol{D}_1$, $\mathcal{S}_{\text{pa1-d-1}}$ is equal to $\mathcal{S}_{\text{cca1-1}}$ (The random tape of $\boldsymbol{C}$ consists of coins for $\boldsymbol{X}$, and the random tape of $\boldsymbol{D}_1$ consists of coins for the encryption algorithm.), and $\mathbf{Exp}^{\text{ind-cca1-1}}_{\mathcal{AE},\boldsymbol{X}}(k)=1$ if and only if $\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1}(k)=1$. ∎

Proof of Claim 4.7: Similarly to the proof of Claim 4.6, it is easy to see that the sample space underlying $\mathbf{Exp}^{\text{ind-cca1-0}}_{\mathcal{AE},\boldsymbol{X}}(k)$ is identical to the sample space underlying $\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0}(k)$, and $\mathbf{Exp}^{\text{ind-cca1-0}}_{\mathcal{AE},\boldsymbol{X}}(k)=1$ if and only if $\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0}(k)=0$. ∎

Proof of Claim 4.8: Let $\mathcal{S}_{\text{pa1-x-1}}$ denote the sample space underlying $\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1,\boldsymbol{C}^*}(k)$. A member of this space is a string specifying the coins of the key-generation algorithm, the random tape of $\boldsymbol{C}$, the random tape of $\boldsymbol{C}^*$, and the random tape of $\boldsymbol{D}_1$. The latter consists of coins for the encryption algorithm.

A member of the sample space $\mathcal{S}_{\text{cpa-1}}$ underlying $\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{AE},\boldsymbol{Y}}(k)$ is a string specifying the coins of the key-generation algorithm, the random tape of $\boldsymbol{Y}$, and the coins used by the encryption algorithm. The random tape of $\boldsymbol{Y}$ consists of coins for $\boldsymbol{C}$ and coins for $\boldsymbol{C}^*$. Hence $\mathcal{S}_{\text{cpa-1}} = \mathcal{S}_{\text{pa1-x-1}}$. We observe that $\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{AE},\boldsymbol{Y}}(k) = 1$ if and only if $\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_1,\boldsymbol{C}^*}(k) = 1$. Claim 4.8 follows. ▌

Proof of Claim 4.9: Similarly to the proof of Claim 4.8, it is easy to see that the sample space underlying $\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{AE},\boldsymbol{Y}}(k)$ is identical to the sample space underlying $\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0,\boldsymbol{C}^*}(k)$, and $\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{AE},\boldsymbol{Y}}(k) = 1$ if and only if $\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}_0,\boldsymbol{C}^*}(k) = 0$. ▌

4.2 Proof of Theorem 4.2

Assume that $\mathcal{AE}$ is PA2 secure and IND-CPA secure, and let $\boldsymbol{X}$ be a polynomial-time ind-cca2-adversary attacking $\mathcal{AE}$. We construct a polynomial-time ciphertext creator $\boldsymbol{C}$ for $\mathcal{AE}$, based on $\boldsymbol{X}$, and polynomial-time plaintext creators $\boldsymbol{P}_0$ and $\boldsymbol{P}_1$, and let $\boldsymbol{C}^*$ be a successful polynomial-time pa2-extractor for $\boldsymbol{C}$. Then we construct a polynomial-time ind-cpa-adversary $\boldsymbol{Y}$ for $\mathcal{AE}$, based on $\boldsymbol{X}$ and $\boldsymbol{C}^*$. Finally, we construct polynomial-time distinguishers $\boldsymbol{D}_0$ and $\boldsymbol{D}_1$ for $\boldsymbol{C}$, and prove that for every $k \in \mathbb{N}$

$$\mathbf{Adv}^{\text{ind-cca2}}_{\mathcal{AE},\boldsymbol{X}}(k) \;\le\; \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE},\boldsymbol{Y}}(k) + \mathbf{Adv}^{\text{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0,\boldsymbol{C}^*}(k) + \mathbf{Adv}^{\text{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1,\boldsymbol{C}^*}(k)\,. \tag{2}$$

The assumption that $\mathcal{AE}$ is PA2 secure and IND-CPA secure implies that $\mathbf{Adv}^{\text{ind-cca2}}_{\mathcal{AE},\boldsymbol{X}}(\cdot)$ is negligible, and thus that $\mathcal{AE}$ is IND-CCA2 secure. We proceed to the constructions and analysis.

The six algorithms we construct are defined in Figure 5. Clearly, they all run in polynomial time. Ciphertext creator $\boldsymbol{C}$ is essentially the same as $\boldsymbol{X}$, except that instead of outputting (M_0, M_1, St), it calls a plaintext creator with argument (M_0, M_1) and, upon receiving a response C, it continues the execution of $\boldsymbol{X}$ by running $\boldsymbol{X}(\mathsf{guess}, \cdot)$ on input C, St. Plaintext creator $\boldsymbol{P}_0$ takes input a pair of messages, and selects the first message. Plaintext creator $\boldsymbol{P}_1$ takes input a pair of messages, and selects the second message. We observe that for $b \in \{0,1\}$, since in experiment $\mathbf{Exp}^{\text{ind-cca2-}b}_{\mathcal{AE},\boldsymbol{X}}(k)$, $\boldsymbol{X}$ does not query its decryption oracle with ciphertext C, in experiments $\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_b,\boldsymbol{D}_b}(k)$ and $\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_b,\boldsymbol{D}_b,\boldsymbol{C}^*}(k)$, $\boldsymbol{C}$ does not make a query (dec, Q) for which $Q \in \textsc{Clist}$. By the assumption that $\mathcal{AE}$ is PA2 secure, there is a successful polynomial-time pa2-extractor $\boldsymbol{C}^*$ for $\boldsymbol{C}$.

A random tape $R[\boldsymbol{C}] \| R[\boldsymbol{C}^*]$ for ind-cpa-adversary $\boldsymbol{Y}$ has two parts, one being a random tape for $\boldsymbol{C}$ (equivalently, for $\boldsymbol{X}$) and the other being a random tape for $\boldsymbol{C}^*$. $\boldsymbol{Y}(\mathsf{find}, \cdot)$ initializes and then maintains state for $\boldsymbol{C}^*$. It runs $\boldsymbol{X}(\mathsf{find}, \cdot)$, and if the latter makes a query, then $\boldsymbol{Y}(\mathsf{find}, \cdot)$ runs $\boldsymbol{C}^*$ to compute a reply, which it returns to $\boldsymbol{X}(\mathsf{find}, \cdot)$. When $\boldsymbol{X}(\mathsf{find}, \cdot)$ outputs (M_0, M_1, St) and stops, $\boldsymbol{Y}(\mathsf{find}, \cdot)$ computes some state information St' that is used by $\boldsymbol{Y}(\mathsf{guess}, \cdot)$ and returns (M_0, M_1, St'). $\boldsymbol{Y}(\mathsf{guess}, \cdot)$ runs $\boldsymbol{X}(\mathsf{guess}, \cdot)$, and if the latter makes a query, then $\boldsymbol{Y}(\mathsf{guess}, \cdot)$ runs $\boldsymbol{C}^*$ to compute a reply, which it returns to $\boldsymbol{X}(\mathsf{guess}, \cdot)$. When $\boldsymbol{X}(\mathsf{guess}, \cdot)$ stops, $\boldsymbol{Y}(\mathsf{guess}, \cdot)$ returns the former's output.

Distinguisher $\boldsymbol{D}_0$ returns the bitwise complement of its input and $\boldsymbol{D}_1$ computes the identity function.

We claim that Equation (2) holds for all $k \in \mathbb{N}$. To prove this, fix $k \in \mathbb{N}$. We state four claims, conclude the proof given them, and then return to prove the claims. The first two claims relate the probability that $\boldsymbol{X}$ guesses the value of challenge bit b, in each of its experiments, to the probability

Ciphertext creator $\boldsymbol{C}(pk; R[\boldsymbol{C}])$
Run $\boldsymbol{X}(\mathsf{find}, \cdot)$ on input pk and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle queries as follows:
– If $\boldsymbol{X}(\mathsf{find}, \cdot)$ makes query Q then
Make query (dec, Q) ; Upon receiving a response M, return M to $\boldsymbol{X}(\mathsf{find}, \cdot)$ as the reply
EndIf
Let (M_0, M_1, St) denote the output of $\boldsymbol{X}(\mathsf{find}, \cdot)$
Make query $(\mathsf{enc}, (M_0, M_1))$; Upon receiving a response C,
run $\boldsymbol{X}(\mathsf{guess}, \cdot)$ on input C, St, until it halts, replying to its oracle queries as follows:
– If $\boldsymbol{X}(\mathsf{guess}, \cdot)$ makes query Q then
Make query (dec, Q) ; Upon receiving a response M, return M to $\boldsymbol{X}(\mathsf{guess}, \cdot)$ as the reply
EndIf
Let d denote the output of $\boldsymbol{X}(\mathsf{guess}, \cdot)$
Return d

Plaintext creator $\boldsymbol{P}_0(Q, \mathrm{St}[\boldsymbol{P}]; R[\boldsymbol{P}])$
Parse Q as (M_0, M_1)
Return $(M_0, \mathrm{St}[\boldsymbol{P}])$

Plaintext creator $\boldsymbol{P}_1(Q, \mathrm{St}[\boldsymbol{P}]; R[\boldsymbol{P}])$
Parse Q as (M_0, M_1)
Return $(M_1, \mathrm{St}[\boldsymbol{P}])$

Adversary $\boldsymbol{Y}(\mathsf{find}, pk; R[\boldsymbol{Y}])$
Parse $R[\boldsymbol{Y}]$ as $R[\boldsymbol{C}] \| R[\boldsymbol{C}^*]$; $\mathrm{St}[\boldsymbol{C}^*] \leftarrow (pk, R[\boldsymbol{C}])$
Run $\boldsymbol{X}(\mathsf{find}, \cdot)$ on input pk and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle queries as follows:
– If $\boldsymbol{X}(\mathsf{find}, \cdot)$ makes query Q then
$(M, \mathrm{St}[\boldsymbol{C}^*]) \leftarrow \boldsymbol{C}^*(Q, \mathrm{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$; Return M to $\boldsymbol{X}(\mathsf{find}, \cdot)$ as the reply EndIf
Let (M_0, M_1, St) denote the output of $\boldsymbol{X}(\mathsf{find}, \cdot)$
$\mathrm{St}' \leftarrow (\mathrm{St}, \mathrm{St}[\boldsymbol{C}^*], R[\boldsymbol{C}^*])$
Return (M_0, M_1, St')

Adversary $\boldsymbol{Y}(\mathsf{guess}, C, \mathrm{St}')$
Parse St' as $(\mathrm{St}, \mathrm{St}[\boldsymbol{C}^*], R[\boldsymbol{C}^*])$
Run $\boldsymbol{X}(\mathsf{guess}, \cdot)$ on input C, St until it halts, replying to its oracle queries as follows:
– If $\boldsymbol{X}(\mathsf{guess}, \cdot)$ makes query Q then
$(M, \mathrm{St}[\boldsymbol{C}^*]) \leftarrow \boldsymbol{C}^*(Q, \mathrm{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$; Return M to $\boldsymbol{X}(\mathsf{guess}, \cdot)$ as the reply EndIf
Let d denote the output of $\boldsymbol{X}(\mathsf{guess}, \cdot)$
Return d

Distinguisher $\boldsymbol{D}_0(x)$
Return $\bar{x}$

Distinguisher $\boldsymbol{D}_1(x)$
Return x

Figure 5: Ciphertext-creator adversary $\boldsymbol{C}$, plaintext-creator adversaries $\boldsymbol{P}_0, \boldsymbol{P}_1$, ind-cpa-adversary $\boldsymbol{Y}$, and distinguishers $\boldsymbol{D}_0, \boldsymbol{D}_1$ for the proof of Theorem 4.2.

---

that distinguisher $\boldsymbol{D}_b$ returns 1 when it is run on the output of $\boldsymbol{C}$ after the latter has interacted with the decryption oracle, in experiment $\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_b,\boldsymbol{D}_b}(k)$.

Claim 4.10 $\Pr\left[\mathbf{Exp}^{\text{ind-cca2-1}}_{\mathcal{AE},\boldsymbol{X}}(k) = 1\right] = \Pr\left[\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1}(k) = 1\right]$. ∎

Claim 4.11 $\Pr\left[\mathbf{Exp}^{\text{ind-cca2-0}}_{\mathcal{AE},\boldsymbol{X}}(k)=1\right] = 1-\Pr\left[\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0}(k)=1\right]$. ∎

The other claims relate the probability that $\boldsymbol{Y}$ guesses the value of challenge bit b, in each of its experiments, to the probability that distinguisher $\boldsymbol{D}_b$ returns 1 when it is run on the output of $\boldsymbol{C}$ after the latter has interacted with pa2-extractor $\boldsymbol{C}^*$, in experiment $\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_b,\boldsymbol{D}_b,\boldsymbol{C}^*}(k)$.

Claim 4.12 $\Pr\left[\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{AE},\boldsymbol{Y}}(k)=1\right] = \Pr\left[\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1,\boldsymbol{C}^*}(k)=1\right]$. ∎

Claim 4.13 $\Pr\left[\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{AE},\boldsymbol{Y}}(k)=1\right] = 1-\Pr\left[\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0,\boldsymbol{C}^*}(k)=1\right]$. ∎

Applying these claims, we obtain Equation (2) as follows:

$$\begin{aligned}
\mathbf{Adv}^{\text{ind-cca2}}_{\mathcal{AE},\boldsymbol{X}}(k) &= \Pr\left[\mathbf{Exp}^{\text{ind-cca2-1}}_{\mathcal{AE},\boldsymbol{X}}(k)=1\right]-\Pr\left[\mathbf{Exp}^{\text{ind-cca2-0}}_{\mathcal{AE},\boldsymbol{X}}(k)=1\right]\\
&= \Pr\left[\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1}(k)=1\right]-\left(1-\Pr\left[\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0}(k)=1\right]\right)\\
&= \left(\Pr\left[\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1,\boldsymbol{C}^*}(k)=1\right]+\mathbf{Adv}^{\text{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1,\boldsymbol{C}^*}(k)\right)-1\\
&\quad+\left(\Pr\left[\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0,\boldsymbol{C}^*}(k)=1\right]+\mathbf{Adv}^{\text{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0,\boldsymbol{C}^*}(k)\right)\\
&= \Pr\left[\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1,\boldsymbol{C}^*}(k)=1\right]-\left(1-\Pr\left[\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0,\boldsymbol{C}^*}(k)=1\right]\right)\\
&\quad+\mathbf{Adv}^{\text{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0,\boldsymbol{C}^*}(k)+\mathbf{Adv}^{\text{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1,\boldsymbol{C}^*}(k)\\
&= \Pr\left[\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{AE},\boldsymbol{Y}}(k)=1\right]-\Pr\left[\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{AE},\boldsymbol{Y}}(k)=1\right]\\
&\quad+\mathbf{Adv}^{\text{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0,\boldsymbol{C}^*}(k)+\mathbf{Adv}^{\text{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1,\boldsymbol{C}^*}(k)\\
&= \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE},\boldsymbol{Y}}(k)+\mathbf{Adv}^{\text{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0,\boldsymbol{C}^*}(k)+\mathbf{Adv}^{\text{pa2}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1,\boldsymbol{C}^*}(k)\ .
\end{aligned}$$

It remains to prove the four claims above.

Proof of Claim 4.10: Let $\mathcal{S}_{\text{cca2-1}}$ denote the sample space underlying $\mathbf{Exp}^{\text{ind-cca2-1}}_{\mathcal{AE},\boldsymbol{X}}(k)$. A member of this space is a string specifying the coins of the key-generation algorithm, the random tape of $\boldsymbol{X}$, and the coins used by the encryption algorithm.

A member of the sample space $\mathcal{S}_{\text{pa2-d-1}}$ underlying $\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1}(k)$ is a string specifying the coins of the key-generation algorithm, the random tape of $\boldsymbol{C}$, the random tape of $\boldsymbol{P}_1$, the coins used by the encryption algorithm across its invocations, and the random tape of $\boldsymbol{D}_1$. Claim 4.10 follows once we observe that by the definitions of $\boldsymbol{C}$, $\boldsymbol{P}_1$ and $\boldsymbol{D}_1$, $\mathcal{S}_{\text{pa2-d-1}}$ is equal to $\mathcal{S}_{\text{cca2-1}}$ (The random tape of $\boldsymbol{C}$ consists of coins for $\boldsymbol{X}$; $\boldsymbol{P}_1$ and $\boldsymbol{D}_1$ are deterministic, so their random tapes have length 0; and in $\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1}(k)$, the encryption algorithm is invoked once.), and $\mathbf{Exp}^{\text{ind-cca2-1}}_{\mathcal{AE},\boldsymbol{X}}(k)=1$ if and only if $\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1}(k)=1$. ∎

Proof of Claim 4.11: Similarly to the proof of Claim 4.10, it is easy to see that the sample space underlying $\mathbf{Exp}^{\text{ind-cca2-0}}_{\mathcal{AE},\boldsymbol{X}}(k)$ is identical to the sample space underlying $\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0}(k)$, and $\mathbf{Exp}^{\text{ind-cca2-0}}_{\mathcal{AE},\boldsymbol{X}}(k)=1$ if and only if $\mathbf{Exp}^{\text{pa2-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0}(k)=0$. ∎

Proof of Claim 4.12: Let $\mathcal{S}_{\text{pa2-x-1}}$ denote the sample space underlying $\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1,\boldsymbol{C}^*}(k)$. A member of this space is a string specifying the coins of the key-generation algorithm, the random tape

of $\boldsymbol{C}$, the random tape of $\boldsymbol{P}_1$, the random tape of $\boldsymbol{C}^*$, the coins used by the encryption algorithm across its invocations, and the random tape of $\boldsymbol{D}_1$. We observe that $\boldsymbol{P}_1$ and $\boldsymbol{D}_1$ are deterministic, so their random tapes have length 0, and that in $\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1,\boldsymbol{C}^*}(k)$, the encryption algorithm is invoked once.

A member of the sample space $\mathcal{S}_{\text{cpa-1}}$ underlying $\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{AE},\boldsymbol{Y}}(k)$ is a string specifying the coins of the key-generation algorithm, the random tape of $\boldsymbol{Y}$, and the coins used by the encryption algorithm. The random tape of $\boldsymbol{Y}$ consists of coins for $\boldsymbol{C}$ and coins for $\boldsymbol{C}^*$. Hence $\mathcal{S}_{\text{cpa-1}} = \mathcal{S}_{\text{pa2-x-1}}$. We observe that $\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{AE},\boldsymbol{Y}}(k) = 1$ if and only if $\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_1,\boldsymbol{D}_1,\boldsymbol{C}^*}(k) = 1$. Claim 4.12 follows. ▌

Proof of Claim 4.13: Similarly to the proof of Claim 4.12, it is easy to see that the sample space underlying $\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{AE},\boldsymbol{Y}}(k)$ is identical to the sample space underlying $\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0,\boldsymbol{C}^*}(k)$, and $\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{AE},\boldsymbol{Y}}(k) = 1$ if and only if $\mathbf{Exp}^{\text{pa2-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{P}_0,\boldsymbol{D}_0,\boldsymbol{C}^*}(k) = 0$. ▌

4.3 Proof of Theorem 4.3

Let $\mathcal{AE}' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ be an IND-CCA2-secure asymmetric encryption scheme. We construct an IND-CCA2-secure asymmetric encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ that is not PA0 secure. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a length preserving one-way function. (This exists assuming IND-CCA2-secure asymmetric encryption schemes exist.) The algorithms constituting $\mathcal{AE}$ are defined as follows:

Algorithm $\mathcal{K}(1^k)$	Algorithm $\mathcal{E}_{pk}(x)$	Algorithm $\mathcal{D}_{sk}(y)$
$(pk', sk') \xleftarrow{\$} \mathcal{K}'(1^k)$	Parse pk as (pk', U)	Parse sk as (sk', u)
$u \xleftarrow{\$} \{0,1\}^k$; $U \leftarrow f(u)$	Return $(0, \mathcal{E}'_{pk'}(x))$	Parse y as (v, y')
$pk \leftarrow (pk', U)$; $sk \leftarrow (sk', u)$		If $v = 0$ then return $\mathcal{D}'_{sk'}(y')$ EndIf
Return (pk, sk)		If $v = 1$ then
		If $y' = f(u)$ then return u
		Else return $\perp$ EndIf EndIf

To prove that $\mathcal{AE}$ is not PA0 secure, we proceed by contradiction. Assume that $\mathcal{AE}$ is PA0 secure and consider the pa0 ciphertext creator $\boldsymbol{C}$ depicted in Figure 6. Notice that $\boldsymbol{C}$ is deterministic and it runs in polynomial time. Let $\boldsymbol{C}^*$ be a successful polynomial-time pa0-extractor for it. We define a polynomial-time distinguisher $\boldsymbol{D}$ for $\boldsymbol{C}$ and a polynomial-time inverter $\boldsymbol{I}$ for function f as shown in Figure 6. Fix $k \in \mathbb{N}$. The probability that $\boldsymbol{I}$ is successful can be computed as follows.

$$\begin{aligned}
\Pr\left[\, U \leftarrow \{0,1\}^k \,;\, u' \xleftarrow{\$} \boldsymbol{I}(U) \;:\; f(u') = U \,\right] &= \Pr\left[\, \mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) = 1 \,\right] \\
&= \Pr\left[\, \mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}}(k) = 1 \,\right] - \mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) \\
&= 1 - \mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)\ .
\end{aligned}$$

Since $\boldsymbol{C}^*$ is a successful pa0-extractor, $\mathbf{Adv}^{\text{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(\cdot)$ is negligible and hence the probability of success of $\boldsymbol{I}$ is not negligible. This contradicts the one-wayness of f, as desired.

We proceed to prove that $\mathcal{AE}$ is IND-CCA2 secure. Let $\boldsymbol{X}$ be an ind-cca2-adversary attacking $\mathcal{AE}$. We define an ind-cca2-adversary $\boldsymbol{X}'$ attacking $\mathcal{AE}'$ as depicted in Figure 6. A random tape $u \| R[\boldsymbol{X}]$ for adversary $\boldsymbol{X}'$ has two parts. The first part is a k-bit string that $\boldsymbol{X}'$ uses to reply to $\boldsymbol{X}$'s queries of the form $(1, f(u))$. (The answer to such a query in experiment $\mathbf{Exp}^{\text{ind-cca2-}b}_{\mathcal{AE},\boldsymbol{X}}(k)$ would be the randomly chosen k-bit string u that corresponds to the second component of the secret key sk.) The second part is a random tape for $\boldsymbol{X}$. $\boldsymbol{X}'$ runs $\boldsymbol{X}$ and uses its decryption oracle and the value u to reply to the oracle

Ciphertext creator $\boldsymbol{C}(pk; R[\boldsymbol{C}])$
Parse pk as (pk', U) ; Make query $(1, U)$; Upon receiving a response u',
Return (pk', U, u')

Distinguisher $\boldsymbol{D}(x)$
Parse x as (pk', U, u') ; If $f(u') = U$ then return 1 else return 0 EndIf

---

Inverter $\boldsymbol{I}(U)$
$k \leftarrow |U|$; $(pk', sk') \xleftarrow{\$} \mathcal{K}'(1^k)$; $pk \leftarrow (pk', U)$
Choose coins $R[\boldsymbol{C}^*]$ for $\boldsymbol{C}^*$; $\mathrm{St}[\boldsymbol{C}^*] \leftarrow (pk, \varepsilon)$; $(u', \mathrm{St}[\boldsymbol{C}^*]) \leftarrow \boldsymbol{C}^*((1, U), \mathrm{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$
Return u'

Adversary $\boldsymbol{X}'(\mathsf{find}, pk'; R[\boldsymbol{X}'])$
Parse $R[\boldsymbol{X}']$ as $u \| R[\boldsymbol{X}]$, where $u \in \{0,1\}^k$; $U \leftarrow f(u)$; $pk \leftarrow (pk', U)$
Run $\boldsymbol{X}(\mathsf{find}, \cdot)$ on input pk and coins $R[\boldsymbol{X}]$ until it halts, replying to its oracle queries as follows:
– If $\boldsymbol{X}(\mathsf{find}, \cdot)$ makes query (v, y') then
If $v = 0$ then
Make query y' ; Upon receiving a response M, return M to $\boldsymbol{X}(\mathsf{find}, \cdot)$ as the reply EndIf
If $v = 1$ then
If $y' = U$ then return u to $\boldsymbol{X}(\mathsf{find}, \cdot)$ as the reply
Else return $\perp$ to $\boldsymbol{X}(\mathsf{find}, \cdot)$ as the reply EndIf EndIf EndIf
Let (M_0, M_1, St) denote the output of $\boldsymbol{X}(\mathsf{find}, \cdot)$
$\mathrm{St}' \leftarrow (\mathrm{St}, u, U)$
Return (M_0, M_1, St')

Adversary $\boldsymbol{X}'(\mathsf{guess}, C', \mathrm{St}')$
Parse St' as (St, u, U) ; $C \leftarrow (0, C')$
Run $\boldsymbol{X}(\mathsf{guess}, \cdot)$ on input C, St until it halts, replying to its oracle queries as follows:
– If $\boldsymbol{X}(\mathsf{guess}, \cdot)$ makes query (v, y') then
If $v = 0$ then
Make query y' ; Upon receiving a response M, return M to $\boldsymbol{X}(\mathsf{guess}, \cdot)$ as the reply EndIf
If $v = 1$ then
If $y' = U$ then return u to $\boldsymbol{X}(\mathsf{guess}, \cdot)$ as the reply
Else return $\perp$ to $\boldsymbol{X}(\mathsf{guess}, \cdot)$ as the reply EndIf EndIf EndIf
Let d denote the output of $\boldsymbol{X}(\mathsf{guess}, \cdot)$
Return d

Figure 6: Ciphertext-creator adversary $\boldsymbol{C}$, distinguisher $\boldsymbol{D}$, inverter $\boldsymbol{I}$, and ind-cca2-adversary $\boldsymbol{X}'$ for the proof of Theorem 4.3.

---

queries of the latter. Clearly, $\boldsymbol{X}'$ runs in polynomial time. Furthermore, for $b \in \{0,1\}$ and $k \in \mathbb{N}$, the replies that experiment $\mathbf{Exp}^{\text{ind-cca2-}b}_{\mathcal{AE}', \boldsymbol{X}'}(k)$ computes to $\boldsymbol{X}'$'s queries allow this adversary to respond to $\boldsymbol{X}$'s queries exactly as experiment $\mathbf{Exp}^{\text{ind-cca2-}b}_{\mathcal{AE}, \boldsymbol{X}}(k)$ does. Therefore, $\mathbf{Adv}^{\text{ind-cca2}}_{\mathcal{AE}', \boldsymbol{X}'}(k) = \mathbf{Adv}^{\text{ind-cca2}}_{\mathcal{AE}, \boldsymbol{X}}(k)$. The assumption that $\mathcal{AE}'$ is IND-CCA2 secure implies that $\mathbf{Adv}^{\text{ind-cca2}}_{\mathcal{AE}', \boldsymbol{X}'}(\cdot)$ is negligible, and thus

Adversary $\boldsymbol{X}(\mathsf{find}, pk; R[\boldsymbol{X}])$	Adversary $\boldsymbol{X}(\mathsf{guess}, C, \mathrm{St})$
Return $(0, 1, \varepsilon)$	Parse C as (r, C') Make query $(\bar{r}, C')$; Let M denote the response If $M = 0$ then $d \leftarrow 0$ else $d \leftarrow 1$ EndIf Return d

Ciphertext creator $\boldsymbol{C}'(pk; R[\boldsymbol{C}'])$

Run $\boldsymbol{C}$ on input pk and coins $R[\boldsymbol{C}']$ until it halts, replying to its oracle queries as follows:

– If $\boldsymbol{C}$ makes query (r, y') then

Make query y' ; Upon receiving a response M, return M to $\boldsymbol{C}$ as the reply EndIf

Let x denote the output of $\boldsymbol{C}$; Return x

pa1-extractor $\boldsymbol{C}^*(Q, \mathrm{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$

Parse Q as (r, y') ; $(M, \mathrm{St}[\boldsymbol{C}'^*]) \leftarrow \boldsymbol{C}'^*(y', \mathrm{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$; Return $(M, \mathrm{St}[\boldsymbol{C}'^*])$

Adversary $\boldsymbol{Y}'(\mathsf{find}, pk; R[\boldsymbol{Y}'])$	Adversary $\boldsymbol{Y}'(\mathsf{guess}, C', \mathrm{St}')$
Parse $R[\boldsymbol{Y}']$ as $r \| R[\boldsymbol{Y}]$, where $r \in \{0,1\}$ $(M_0, M_1, \mathrm{St}) \xleftarrow{\$} \boldsymbol{Y}(\mathsf{find}, pk; R[\boldsymbol{Y}])$; $\mathrm{St}' \leftarrow (\mathrm{St}, r)$ Return (M_0, M_1, St')	Parse St' as (St, r) ; $C \leftarrow (r, C')$ $d \leftarrow \boldsymbol{Y}(\mathsf{guess}, C, \mathrm{St})$ Return d

Figure 7: ind-cca2-adversary $\boldsymbol{X}$, ciphertext-creator adversary $\boldsymbol{C}'$, pa1-extractor $\boldsymbol{C}^*$, and ind-cpa-adversary $\boldsymbol{Y}'$ for the proof of Theorem 4.4.

$\mathbf{Adv}^{\text{ind-cca2}}_{\mathcal{AE},\boldsymbol{X}}(\cdot)$ is negligible. Hence $\mathcal{AE}$ is IND-CCA2 secure.

4.4 Proof of Theorem 4.4

Let $\mathcal{AE}' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ be a PA1 secure and IND-CPA-secure asymmetric encryption scheme. We construct a PA1 secure and IND-CPA-secure asymmetric encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ that is not IND-CCA2 secure. Notice that the key-generation algorithm is the same. The encryption and decryption algorithms are defined as follows:

Algorithm $\mathcal{E}_{pk}(x)$	Algorithm $\mathcal{D}_{sk}(y)$
$r \xleftarrow{\$} \{0,1\}$ Return $(r, \mathcal{E}'_{pk}(x))$	Parse y as (r, y') ; $x \leftarrow \mathcal{D}'_{sk}(y')$ Return x

To prove that $\mathcal{AE}$ is not IND-CCA2 secure, we define an ind-cca2-adversary $\boldsymbol{X}$ attacking $\mathcal{AE}$ as shown in Figure 7. Clearly, $\boldsymbol{X}$ runs in polynomial time and $\mathbf{Adv}^{\text{ind-cca2}}_{\mathcal{AE},\boldsymbol{X}}(k) = 1$ for every $k \in \mathbb{N}$.

To prove that $\mathcal{AE}$ is PA1 secure, let $\boldsymbol{C}$ be a polynomial-time ciphertext creator attacking $\mathcal{AE}$. We define a ciphertext creator $\boldsymbol{C}'$ attacking $\mathcal{AE}'$ as shown in Figure 7. Clearly, $\boldsymbol{C}'$ runs in polynomial time. By the assumption that $\mathcal{AE}'$ is PA1 secure, there is a successful polynomial-time pa1-extractor $\boldsymbol{C}'^*$ for $\boldsymbol{C}'$. We construct a pa1-extractor $\boldsymbol{C}^*$ for $\boldsymbol{C}$, based on $\boldsymbol{C}'^*$ as shown in Figure 7. It is clear that $\boldsymbol{C}^*$ runs in polynomial time. Let $\boldsymbol{D}$ be a polynomial-time distinguisher for $\boldsymbol{C}$, and fix $k \in \mathbb{N}$. It is easy to see that

$$\Pr\left[\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}}(k) = 1\right] = \Pr\left[\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE}',\boldsymbol{C}',\boldsymbol{D}}(k) = 1\right] \quad \text{and}$$

$$\Pr\left[\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) = 1\right] = \Pr\left[\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE}',\boldsymbol{C}',\boldsymbol{D},\boldsymbol{C}'^*}(k) = 1\right].$$

Therefore,

$$\mathbf{Adv}^{\mathrm{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) \;=\; \mathbf{Adv}^{\mathrm{pa1}}_{\mathcal{AE}',\boldsymbol{C}',\boldsymbol{D},\boldsymbol{C}'^*}(k)\;.$$

Since $\boldsymbol{C}'^*$ is a successful pa1-extractor for $\boldsymbol{C}'$, for every polynomial-time distinguisher $\boldsymbol{D}$, the function $\mathbf{Adv}^{\mathrm{pa1}}_{\mathcal{AE}',\boldsymbol{C}',\boldsymbol{D},\boldsymbol{C}'^*}(\cdot)$ is negligible and hence for every polynomial-time distinguisher $\boldsymbol{D}$, the function $\mathbf{Adv}^{\mathrm{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(\cdot)$ is negligible. Thus $\boldsymbol{C}^*$ is a successful pa1-extractor for $\boldsymbol{C}$, and $\mathcal{AE}$ is PA1 secure.

To prove that $\mathcal{AE}$ is IND-CPA secure, let $\boldsymbol{Y}$ be an ind-cpa-adversary attacking $\mathcal{AE}$. Consider the ind-cpa-adversary $\boldsymbol{Y}'$ attacking $\mathcal{AE}'$ depicted in Figure 7. A random tape $r\|R[\boldsymbol{Y}]$ for adversary $\boldsymbol{Y}'$ has two parts. The first part is a bit that $\boldsymbol{Y}'$ uses to compute the challenge ciphertext C for $\boldsymbol{Y}$. The second part is a random tape for $\boldsymbol{Y}$. $\boldsymbol{Y}'$ runs $\boldsymbol{Y}$ and returns the output of the latter. Clearly, $\boldsymbol{Y}'$ runs in polynomial time and $\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE}',\boldsymbol{Y}'}(k) = \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE},\boldsymbol{Y}}(k)$ for every $k \in \mathbb{N}$. The assumption that $\mathcal{AE}'$ is IND-CPA secure implies that the ind-cpa-advantage of $\boldsymbol{Y}'$ is negligible, and hence it follows that the ind-cpa-advantage of $\boldsymbol{Y}$ is negligible. Thus $\mathcal{AE}$ is IND-CPA secure.

4.5 Proof of Theorem 4.5

Let $\mathcal{AE}' = (\mathcal{K}, \mathcal{E}', \mathcal{D}')$ be a PA0 secure and IND-CPA-secure asymmetric encryption scheme. We construct a PA0 secure and IND-CPA-secure asymmetric encryption scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ that is not IND-CCA1 secure. Its constituent algorithms are defined as follows:

Algorithm $\mathcal{K}(1^k)$
  $(pk', sk') \xleftarrow{\$} \mathcal{K}'(1^k)$
  $u \xleftarrow{\$} \{0,1\}^{|sk'|}$; $sk \leftarrow (sk', u)$
  Return (pk', sk)

Algorithm $\mathcal{E}_{pk}(x)$
  Return $(0, \mathcal{E}'_{pk}(x))$

Algorithm $\mathcal{D}_{sk}(y)$
  Parse sk as (sk', u)
  Parse y as (v, y')
  If $v = 0$ then return $\mathcal{D}'_{sk'}(y')$ EndIf
  If $v = 1$ then
    If $y' = 0$ then return u
    Else return $u \oplus sk'$ EndIf EndIf

To prove that $\mathcal{AE}$ is not IND-CCA1 secure, we define an ind-cca1-adversary $\boldsymbol{X}$ attacking $\mathcal{AE}$ as shown in Figure 8. Clearly, $\boldsymbol{X}$ runs in polynomial time and $\mathbf{Adv}^{\text{ind-cca1}}_{\mathcal{AE},\boldsymbol{X}}(k) = 1$ for every $k \in \mathbb{N}$.

To prove that $\mathcal{AE}$ is PA0 secure, let $\boldsymbol{C}$ be a polynomial-time pa0 ciphertext creator attacking $\mathcal{AE}$. We define a pa0 ciphertext creator $\boldsymbol{C}'$ attacking $\mathcal{AE}'$ as shown in Figure 8. Clearly, $\boldsymbol{C}'$ runs in polynomial time. By the assumption that $\mathcal{AE}'$ is PA0 secure, there is a successful polynomial-time pa0-extractor $\boldsymbol{C}'^*$ for $\boldsymbol{C}'$. We construct a pa0-extractor $\boldsymbol{C}^*$ for $\boldsymbol{C}$, based on $\boldsymbol{C}'^*$ as shown in Figure 8. It is clear that $\boldsymbol{C}^*$ runs in polynomial time. Let $\boldsymbol{D}$ be a polynomial-time distinguisher for $\boldsymbol{C}$, and fix $k \in \mathbb{N}$. It is easy to see that

$$\begin{aligned}
\Pr\left[\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}}(k) = 1\right] &= \Pr\left[\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE}',\boldsymbol{C}',\boldsymbol{D}}(k) = 1\right] \quad \text{and} \\
\Pr\left[\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) = 1\right] &= \Pr\left[\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE}',\boldsymbol{C}',\boldsymbol{D},\boldsymbol{C}'^*}(k) = 1\right].
\end{aligned}$$

Therefore,

$$\mathbf{Adv}^{\mathrm{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) \;=\; \mathbf{Adv}^{\mathrm{pa1}}_{\mathcal{AE}',\boldsymbol{C}',\boldsymbol{D},\boldsymbol{C}'^*}(k)\;.$$

Since $\boldsymbol{C}'^*$ is a successful pa0-extractor for $\boldsymbol{C}'$, for every polynomial-time distinguisher $\boldsymbol{D}$, the function $\mathbf{Adv}^{\mathrm{pa1}}_{\mathcal{AE}',\boldsymbol{C}',\boldsymbol{D},\boldsymbol{C}'^*}(\cdot)$ is negligible and hence for every polynomial-time distinguisher $\boldsymbol{D}$, the function $\mathbf{Adv}^{\mathrm{pa1}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(\cdot)$ is negligible. Thus $\boldsymbol{C}^*$ is a successful pa0-extractor for $\boldsymbol{C}$, and $\mathcal{AE}$ is PA0 secure.

To prove that $\mathcal{AE}$ is IND-CPA secure, let $\boldsymbol{Y}$ be an ind-cpa-adversary attacking $\mathcal{AE}$. Consider the ind-cpa-adversary $\boldsymbol{Y}'$ attacking $\mathcal{AE}'$ depicted in Figure 8. Clearly, $\boldsymbol{Y}'$ runs in polynomial time and

Adversary $\boldsymbol{X}(\mathsf{find}, pk; R[\boldsymbol{X}])$	Adversary $\boldsymbol{X}(\mathsf{guess}, C, \mathrm{St})$
Make query $(1,0)$; Let M_0 denote the response Make query $(1,1)$; Let M_1 denote the response $sk' \leftarrow M_0 \oplus M_1$; Return $(0,1,sk')$	Parse C as $(0,C')$ If $\mathcal{D}'_{\mathrm{St}}(C') = 0$ then $d \leftarrow 0$ else $d \leftarrow 1$ EndIf Return d

Ciphertext creator $\boldsymbol{C}'(pk; R[\boldsymbol{C}'])$
Parse $R[\boldsymbol{C}']$ as $u \| R[\boldsymbol{C}]$
Run $\boldsymbol{C}$ on input pk and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle query as follows:
– When $\boldsymbol{C}$ makes query (v, y') do
Make query y' ; Let M denote the response ; If $v = 1$ then $M \leftarrow u$ EndIf
Return M to $\boldsymbol{C}$ as the reply
Let x denote the output of $\boldsymbol{C}$; Return x

pa0-extractor $\boldsymbol{C}^*(Q, \mathrm{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$
Parse $\mathrm{St}[\boldsymbol{C}^*]$ as $(pk, R[\boldsymbol{C}])$; Parse $R[\boldsymbol{C}^*]$ as $u \| R[\boldsymbol{C}'^*]$; Parse Q as (v, y')
$\mathrm{St}[\boldsymbol{C}'^*] \leftarrow (pk, u \| R[\boldsymbol{C}])$; $(M, \mathrm{St}[\boldsymbol{C}'^*]) \leftarrow \boldsymbol{C}'^*(y', \mathrm{St}[\boldsymbol{C}'^*]; R[\boldsymbol{C}'^*])$; If $v = 1$ then $M \leftarrow u$ EndIf
Return $(M, \mathrm{St}[\boldsymbol{C}'^*])$

Adversary $\boldsymbol{Y}'(\mathsf{find}, pk; R[\boldsymbol{Y}'])$	Adversary $\boldsymbol{Y}'(\mathsf{guess}, C', \mathrm{St})$
$(M_0, M_1, \mathrm{St}) \xleftarrow{\$} \boldsymbol{Y}(\mathsf{find}, pk; R[\boldsymbol{Y}'])$ Return (M_0, M_1, St)	$C \leftarrow (0, C')$; $d \leftarrow \boldsymbol{Y}(\mathsf{guess}, C, \mathrm{St})$ Return d

Figure 8: ind-cca1-adversary $\boldsymbol{X}$, pa0 ciphertext-creator adversary $\boldsymbol{C}'$, pa0-extractor $\boldsymbol{C}^*$, and ind-cpa-adversary $\boldsymbol{Y}'$ for the proof of Theorem 4.5.

$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE}',\boldsymbol{Y}'}(k) = \mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{AE},\boldsymbol{Y}}(k)$ for every $k \in \mathbb{N}$. The assumption that $\mathcal{AE}'$ is IND-CPA secure implies that the ind-cpa-advantage of $\boldsymbol{Y}'$ is negligible, and hence it follows that the ind-cpa-advantage of $\boldsymbol{Y}$ is negligible. Thus $\mathcal{AE}$ is IND-CPA secure.

5 Constructions

Approaches. Before presenting our results, we discuss some possible approaches to designing PA encryption schemes. One natural approach is based on the use of non-interactive zero-knowledge proofs of knowledge (NIZK-POKs) [15]. In particular, a candidate construction is the following. Let the public key have the form (pk, R) where pk is the public key of some IND-CPA (or even IND-CCA2) "base" encryption scheme and R is a random reference string. Encryption consists of providing an encryption of the message under pk via the base scheme, together with a NIZK-POK of the message relative to reference string R. However, this type of approach fails to yield even the weakest form of PA. The problem is that the PA extractor must work with the given public key of the ciphertext creator, and hence a given reference string, while the NIZK-POK extractor that one would hope to use to define the PA extractor, creates a simulated reference string with accompanying trapdoor.

This might lead one to ask why our definition of PA is not relaxed to allow the extractor to choose or simulate the public key rather than having to work with the given one. Besides the fact that the intuition captured is quite different, it is not clear how to make such a relaxation while preserving

Experiment $\mathbf{Exp}^{\text{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(k)$

$(p,q,g) \xleftarrow{\$} G(1^k)$; $a \xleftarrow{\$} \mathbb{Z}_q$; $A \leftarrow g^a \bmod p$

Choose coins $R[\boldsymbol{H}], R[\boldsymbol{H}^*]$ for $\boldsymbol{H}, \boldsymbol{H}^*$, respectively ; $\text{St}[\boldsymbol{H}^*] \leftarrow ((p,q,g,A), R[\boldsymbol{H}])$

Run $\boldsymbol{H}$ on input p,q,g,A and coins $R[\boldsymbol{H}]$ until it halts, replying to its oracle queries as follows:

– If $\boldsymbol{H}$ makes query (B,W) then

$(b, \text{St}[\boldsymbol{H}^*]) \leftarrow \boldsymbol{H}^*((B,W), \text{St}[\boldsymbol{H}^*]; R[\boldsymbol{H}^*])$

If $W \equiv B^a \pmod p$ and $B \not\equiv g^b \pmod p$ then return 1

Else return b to $\boldsymbol{H}$ as the reply EndIf EndIf

Return 0

Figure 9: Experiment used to define the DHK1 and DHK0 assumptions.

---

the PA1+IND-CPA → IND-CCA1 and PA2+IND-CPA → IND-CCA2 implications of Theorems 4.1 and 4.2. (In particular, if we allow the extractor to simply choose a public key, it can choose one whose corresponding secret key it knows, making the notion trivial to achieve and making the implications fail.)

As such approaches are considered and discarded, it becomes apparent that achieving even the weaker forms of PA under standard assumptions may be difficult. We have been able to make progress, however, under some strong assumptions, as we now describe.

PRIME-ORDER GROUPS. If p,q are primes such that $p = 2q+1$, then we let G_q denote the subgroup of quadratic residues of $\mathbb{Z}_p^*$. Recall this is a cyclic subgroup of order q. If g is a generator of G_q then $\text{dlog}_{q,g}(X)$ denotes the discrete logarithm of $X \in G_q$ to base g. A *prime-order-group generator* is a polynomial-time algorithm G that on input 1^k returns a triple (p,q,g) such that p,q are primes with $p = 2q+1$, g is a generator of G_q, and $2^{k-1} < p < 2^k$ (p is k bits long).

THE DHK ASSUMPTIONS. Let G be a prime-order-group generator, and suppose $(p,q,g) \in [G(1^k)]$. We say that (A,B,W) is a *DH-triple* if there exist $a,b \in Z_q$ such that $A = g^a \bmod p$, $B = g^b \bmod p$ and $W = g^{ab} \bmod p$. We say that (B,W) is a *DH-pair relative to* A if (A,B,W) is a DH-triple. One way for an adversary $\boldsymbol{H}$ taking input p,q,g,A to output a DH-pair (B,W) relative to A is to pick — and thus "know"— some $b \in \mathbb{Z}_q$, set $B = g^b \bmod p$ and $W = A^b \bmod p$, and output (B,W). Damgård [14] makes an assumption which, informally, says that this is the "only" way that a polynomial-time adversary $\boldsymbol{H}$ can output a DH-pair relative to A. His framework to capture this requires that there exist a suitable extractor $\boldsymbol{H}^*$ that can compute $\text{dlog}_{q,g}(B)$ whenever $\boldsymbol{H}$ outputs some DH-pair (B,W) relative to A.

We provide a formalization of this assumption that we refer to as the DHK0 (DHK stands for Diffie-Hellman Knowledge) assumption. We also present a natural extension of this assumption that we refer to as DHK1. Here the adversary $\boldsymbol{H}$, given p,q,g,A, interacts with the extractor, querying it adaptively. The extractor is required to be able to return $\text{dlog}_{q,g}(B)$ for each DH-pair (B,W) relative to A that is queried to it. Below we first present the DHK1 assumption, and then define the DHK0 assumption via this.

Assumption 5.1 [**DHK1**] Let G be a prime-order-group generator. Let $\boldsymbol{H}$ be an algorithm that has access to an oracle, takes two primes and two group elements, and returns nothing. Let $\boldsymbol{H}^*$ be an algorithm that takes a pair of group elements and some state information, and returns an exponent and a new state. We call $\boldsymbol{H}$ a dhk1-*adversary* and $\boldsymbol{H}^*$ a dhk1-*extractor*. For $k \in \mathbb{N}$ we define the

Algorithm $\mathcal{K}(1^k)$	Algorithm $\mathcal{E}((p,q,g,X_1,X_2),M)$
$(p,q,g) \xleftarrow{\$} G(1^k)$ $x_1 \xleftarrow{\$} \mathbb{Z}_q$; $X_1 \leftarrow g^{x_1} \bmod p$ $x_2 \xleftarrow{\$} \mathbb{Z}_q$; $X_2 \leftarrow g^{x_2} \bmod p$ Return $((p,q,g,X_1,X_2),(p,q,g,x_1,x_2))$	$y \xleftarrow{\$} \mathbb{Z}_q$; $Y \leftarrow g^y \bmod p$ $W \leftarrow X_1^y \bmod p$; $V \leftarrow X_2^y \bmod p$ $U \leftarrow V \cdot M \bmod p$ Return (Y,W,U)
Algorithm $\mathcal{D}((p,q,g,x_1,x_2),(Y,W,U))$ If $W \not\equiv Y^{x_1} \pmod p$ then return $\perp$ Else $M \leftarrow U \cdot Y^{-x_2} \bmod p$; Return M EndIf	$\mathrm{MsgSp}((p,q,g,X_1,X_2)) = G_q$

Figure 10: Algorithms of the encryption scheme DEG $= (\mathcal{K},\mathcal{E},\mathcal{D},\mathrm{MsgSp})$ based on prime-order-group generator G.

experiment shown in Figure 9. The dhk1-*advantage* of $\boldsymbol{H}$ relative to $\boldsymbol{H}^*$ is

$$\mathbf{Adv}^{\mathrm{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(k) \;=\; \Pr\left[\,\mathbf{Exp}^{\mathrm{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(k)=1\,\right].$$

We say that G satisfies the DHK1 assumption if for every polynomial-time dhk1-adversary $\boldsymbol{H}$ there exists a polynomial-time dhk1-extractor $\boldsymbol{H}^*$ such that $\mathbf{Adv}^{\mathrm{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(\cdot)$ is negligible.

Assumption 5.2 [**DHK0**] Let G be a prime-order-group generator. We call a dhk1-adversary that makes *exactly one* oracle query a dhk0-*adversary*. We call a dhk1-extractor for a dhk0-adversary a dhk0-*extractor*. We say that G satisfies the Diffie-Hellman Knowledge (DHK0) assumption if for every polynomial-time dhk0-adversary $\boldsymbol{H}$ there exists a polynomial-time dhk0-extractor $\boldsymbol{H}^*$ such that $\mathbf{Adv}^{\mathrm{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(\cdot)$ is negligible.

We observe that DHK1 implies DHK0 in the sense that if a prime-order-group generator satisfies the former assumption then it also satisfies the latter assumption.

Constructions. We would like to build an asymmetric encryption scheme that is PA0 secure (and IND-CPA secure) under the DHK0 assumption. An obvious idea is to use ElGamal encryption. Here the public key is $X = g^x$, where x is the secret key, and an encryption of message $M \in G_q$ has the form (Y,U), where $Y = g^y \bmod p$ and $U = X^y \cdot M \bmod p = g^{xy} \cdot M \bmod p$. However, we do not know whether this scheme is PA0 secure. (We can show that it is not sPA0 secure unless the discrete-logarithm problem is easy, but whether or not it is PA0 remains open.)

We consider a modification of the ElGamal scheme that was proposed by Damgård [14]. We call this scheme *Damgård ElGamal* or DEG. It is parameterized by a prime-order group generator G, and its components are depicted in Figure 10. The proof of the following is in Section 5.2:

Theorem 5.3 Let G be a prime-order-group generator and let DEG $= (\mathcal{K},\mathcal{E},\mathcal{D},\mathrm{MsgSp})$ be the associated Damgård ElGamal asymmetric encryption scheme defined in Figure 10. If G satisfies the DHK0 and DDH assumptions then DEG is PA0+IND-CPA secure. If G satisfies the DHK1 and DDH assumptions then DEG is PA1+IND-CPA secure. ▌

As a consequence of the above and Theorem 4.1, DEG is IND-CCA1 secure under the DHK1 and DDH assumptions. DEG is in fact the most efficient known IND-CCA1 scheme with some proof of security in the standard model.

Next we consider the "lite" version of the Cramer-Shoup asymmetric encryption scheme [13]. The scheme, denoted CS-lite, is parameterized by a prime-order group generator G, and its components

Algorithm $\mathcal{K}(1^k)$	Algorithm $\mathcal{E}((p,q,g_1,g_2,X,Z),M)$
$(p,q,g_1) \xleftarrow{\$} G(1^k)$; $g_2 \xleftarrow{\$} G_q \setminus \{1\}$ $x_1 \xleftarrow{\$} \mathbb{Z}_q$; $x_2 \xleftarrow{\$} \mathbb{Z}_q$; $z \xleftarrow{\$} \mathbb{Z}_q$ $X \leftarrow g_1^{x_1} \cdot g_2^{x_2} \bmod p$; $Z \leftarrow g_1^z \bmod p$ Return $((p,q,g_1,g_2,X,Z),(p,q,g_1,g_2,x_1,x_2,z))$	$r \xleftarrow{\$} \mathbb{Z}_q$ $R_1 \leftarrow g_1^r \bmod p$; $R_2 \leftarrow g_2^r \bmod p$ $E \leftarrow Z^r \cdot M \bmod p$; $V \leftarrow X^r \bmod p$ Return (R_1,R_2,E,V)
Algorithm $\mathcal{D}((p,q,g_1,g_2,x_1,x_2,z),(R_1,R_2,E,V))$	$\mathrm{MsgSp}((p,q,g_1,g_2,X,Z)) = G_q$
If $V \not\equiv R_1^{x_1} \cdot R_2^{x_2} \pmod p$ then return $\perp$ Else $M \leftarrow E \cdot R_1^{-z} \bmod p$; Return M EndIf	

Figure 11: Algorithms of the encryption scheme CS-lite $= (\mathcal{K},\mathcal{E},\mathcal{D},\mathrm{MsgSp})$ based on prime-order-group generator G.

are depicted in Figure 11. This scheme is known to be IND-CCA1 secure under the DDH assumption [13]. We are able to show the following. The proof can be found in Section 5.3:

Theorem 5.4 Let G be a prime-order-group generator, and let CS-lite $= (\mathcal{K},\mathcal{E},\mathcal{D},\mathrm{MsgSp})$ be the associated Cramer-Shoup lite asymmetric encryption scheme defined in Figure 11. If G satisfies the DHK0 and DDH assumptions then CS-lite is PA0+IND-CPA secure. If G satisfies the DHK1 and DDH assumptions then CS-lite is PA1+IND-CPA secure. ▌

Again, the above and Theorem 4.1 imply that CS-lite is IND-CCA1 secure under the DHK1 and DDH assumptions. This however is not news, since we already know that DDH alone suffices to prove it IND-CCA1 [13]. However, it does perhaps provide a new perspective on why the scheme is IND-CCA1, namely that this is due to its possessing some form of plaintext awareness.

In summary, we have been able to show that plaintext awareness without ROs is efficiently achievable, even though under very strong and non-standard assumptions.

5.1 A lemma

We first state and prove a lemma that will be used in the proofs of the theorems stated above.

Lemma 5.5 Let $\mathcal{AE}$ be an asymmetric encryption scheme. Let $\boldsymbol{C}$ be a polynomial-time ciphertext creator attacking $\mathcal{AE}$, $\boldsymbol{D}$ a polynomial-time distinguisher, and $\boldsymbol{C}^*$ a polynomial-time pa1-extractor. Let DecOK denote the event that all $\boldsymbol{C}^*$'s answers to $\boldsymbol{C}$'s queries are correct in experiment $\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)$. Then,

$$\Pr\left[\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)=1\right] \;\geq\; \Pr\left[\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}}(k)=1\right] - \Pr\left[\overline{\text{DecOK}}\right]. \qquad ▌$$

Proof: We observe that if experiment $\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)$ returns 0 and event DecOK occurs, then experiment $\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}}(k)$ also returns 0. Therefore,

$$\begin{aligned}
1-\Pr\left[\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)=1\right] &= \Pr\left[\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)=0\right] \\
&= \Pr\left[\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)=0 \wedge \text{DecOK}\right] + \Pr\left[\mathbf{Exp}^{\text{pa1-x}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)=0 \wedge \overline{\text{DecOK}}\right] \\
&\leq \Pr\left[\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}}(k)=0\right] + \Pr\left[\overline{\text{DecOK}}\right] = 1-\Pr\left[\mathbf{Exp}^{\text{pa1-d}}_{\mathcal{AE},\boldsymbol{C},\boldsymbol{D}}(k)=1\right] + \Pr\left[\overline{\text{DecOK}}\right]
\end{aligned}$$

dhk1-adversary $\boldsymbol{H}(p, q, g, A; R[\boldsymbol{H}])$
Parse $R[\boldsymbol{H}]$ as $X_2 \| R[\boldsymbol{C}]$ where $X_2 \in G_q$
Run $\boldsymbol{C}$ on input (p, q, g, A, X_2) and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle queries as follows:
– If $\boldsymbol{C}$ makes query (Y, W, U) then
Make query (Y, W) ; Let b denote the response
If $\big(Y \not\equiv g^b \pmod p$ or $W \not\equiv A^b \pmod p\big)$ then $M \leftarrow \perp$ else $M \leftarrow U \cdot X_2^{-b} \bmod p$ EndIf
Return M to $\boldsymbol{C}$ as the reply EndIf
Halt

---

pa1-extractor $\boldsymbol{C}^*(Q, \mathrm{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$
If $\mathrm{St}[\boldsymbol{C}^*]$ is the initial state then
Parse $\mathrm{St}[\boldsymbol{C}^*]$ as $((p, q, g, A, X_2), R[\boldsymbol{C}])$; $\mathrm{St}[\boldsymbol{H}^*] \leftarrow ((p, q, g, A), X_2 \| R[\boldsymbol{C}])$
Else Parse $\mathrm{St}[\boldsymbol{C}^*]$ as $((p, q, g, A, X_2), \mathrm{St}[\boldsymbol{H}^*])$ EndIf
Parse Q as (Y, W, U) ; $(b, \mathrm{St}[\boldsymbol{H}^*]) \leftarrow \boldsymbol{H}^*((Y, W), \mathrm{St}[\boldsymbol{H}^*]; R[\boldsymbol{C}^*])$
If $\big(Y \not\equiv g^b \pmod p$ or $W \not\equiv A^b \pmod p\big)$ then $M \leftarrow \perp$ else $M \leftarrow U \cdot X_2^{-b} \bmod p$ EndIf
$\mathrm{St}[\boldsymbol{C}^*] \leftarrow ((p, q, g, A, X_2), \mathrm{St}[\boldsymbol{H}^*])$
Return $(M, \mathrm{St}[\boldsymbol{C}^*])$

Adversary $\boldsymbol{Y}'(\mathsf{find}, (p, q, g, X); R[\boldsymbol{Y}'])$
Parse $R[\boldsymbol{Y}']$ as $x_1 \| R[\boldsymbol{Y}]$, where $x_1 \in \mathbb{Z}_q$; $X_1 \leftarrow g^{x_1} \bmod p$
$(M_0, M_1, \mathrm{St}) \xleftarrow{\$} \boldsymbol{Y}(\mathsf{find}, (p, q, g, X_1, X); R[\boldsymbol{Y}])$; $\mathrm{St}' \leftarrow (\mathrm{St}, x_1)$
Return (M_0, M_1, St')

Adversary $\boldsymbol{Y}'(\mathsf{guess}, C', \mathrm{St}')$
Parse St' as (St, x_1) ; Parse C' as (Y, U) ; $W \leftarrow Y^{x_1} \bmod p$; $C \leftarrow (Y, W, U)$
$d \leftarrow \boldsymbol{Y}(\mathsf{guess}, C, \mathrm{St})$; Return d

Figure 12: dhk1-adversary $\boldsymbol{H}$, pa1-extractor $\boldsymbol{C}^*$, and ind-cpa-adversary $\boldsymbol{Y}'$ for the proof of Theorem 5.3.

Transposing terms and simplifying completes the proof of the lemma. ∎

5.2 Proof of Theorem 5.3

We first show that the DHK1 assumption implies DEG is PA1 secure, and then that the DDH assumption implies it is IND-CPA secure. Finally we briefly indicate how to show that the DHK0 assumption implies DEG is PA0 secure.

Let $\boldsymbol{C}$ be a polynomial-time ciphertext creator attacking DEG. We build a polynomial-time pa1-extractor $\boldsymbol{C}^*$ for it. To do so, we first define a polynomial-time dhk1-adversary $\boldsymbol{H}$ attacking prime-order-group generator G. By the DHK1 assumption, $\boldsymbol{H}$ has a polynomial-time dhk1-extractor $\boldsymbol{H}^*$. We then use $\boldsymbol{H}^*$ to build $\boldsymbol{C}^*$. The descriptions of $\boldsymbol{H}$ and $\boldsymbol{C}^*$ are in Figure 12.

The random tape $X_2 \| R[\boldsymbol{C}]$ of $\boldsymbol{H}$ consists of a choice X_2 of an element in the group G_q together with a random tape $R[\boldsymbol{C}]$ for $\boldsymbol{C}$. The random tape of pa1-extractor $\boldsymbol{C}^*$ consists of a random tape for dhk1-extractor $\boldsymbol{H}^*$. Observe that extractor $\boldsymbol{C}^*$ gets input the random tape $R[\boldsymbol{C}]$ of $\boldsymbol{C}$, while extractor $\boldsymbol{H}^*$ must get as input the random tape $R[\boldsymbol{H}] = X_2 \| R[\boldsymbol{C}]$ of $\boldsymbol{H}$. Clearly, $\boldsymbol{H}$ and $\boldsymbol{C}^*$ are

polynomial time. We claim that $\boldsymbol{C}^*$ is a successful pa1-extractor for $\boldsymbol{C}$. To prove this, let $\boldsymbol{D}$ be a polynomial-time distinguisher for $\boldsymbol{C}$, and fix $k \in \mathbb{N}$. We state a claim, conclude the proof given this claim and Lemma 5.5, and then return to prove the claim.

Claim 5.6 Let DECOK denote the event that all $\boldsymbol{C}^*$'s answers to $\boldsymbol{C}$'s queries are correct in experiment $\mathbf{Exp}^{\text{pa1-x}}_{\text{DEG},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)$. Then, $\Pr\left[\,\overline{\text{DECOK}}\,\right] \leq \mathbf{Adv}^{\text{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(k)$. ▌

Applying Lemma 5.5 and Claim 5.6, we obtain the desired result as follows.

$$\begin{aligned}
\mathbf{Adv}^{\text{pa1}}_{\text{DEG},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) &= \Pr\left[\mathbf{Exp}^{\text{pa1-d}}_{\text{DEG},\boldsymbol{C},\boldsymbol{D}}(k)=1\right] - \Pr\left[\mathbf{Exp}^{\text{pa1-x}}_{\text{DEG},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)=1\right] \\
&\leq \Pr\left[\mathbf{Exp}^{\text{pa1-d}}_{\text{DEG},\boldsymbol{C},\boldsymbol{D}}(k)=1\right] - \Pr\left[\mathbf{Exp}^{\text{pa1-d}}_{\text{DEG},\boldsymbol{C},\boldsymbol{D}}(k)=1\right] + \Pr\left[\,\overline{\text{DECOK}}\,\right] \\
&\leq \mathbf{Adv}^{\text{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(k)
\end{aligned}$$

By the DHK1 assumption, the function $\mathbf{Adv}^{\text{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(\cdot)$ is negligible and hence for every polynomial-time distinguisher $\boldsymbol{D}$, the function $\mathbf{Adv}^{\text{pa1}}_{\text{DEG},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(\cdot)$ is negligible. Thus $\boldsymbol{C}^*$ is a successful pa1-extractor for $\boldsymbol{C}$, and DEG is PA1 secure. It remains to prove the claim above.

Proof of Claim 5.6: We observe that by the definition of pa1-extractor $\boldsymbol{C}^*$ and DEG's decryption algorithm $\mathcal{D}$, if $\boldsymbol{C}^*$'s response M to a query (Y,W,U) made by $\boldsymbol{C}$ is such that $M \neq \perp$ then $\mathcal{D}((p,q,g,x_1,x_2),(Y,W,U)) \neq \perp$ and $M = \mathcal{D}((p,q,g,x_1,x_2),(Y,W,U))$. Therefore,

$$\begin{aligned}
\Pr\left[\,\overline{\text{DECOK}}\,\right] &= \Pr[\boldsymbol{C} \text{ makes a query } (Y,W,U) \text{ for which } \boldsymbol{C}^*\text{'s response } M \text{ is such that} \\
&\qquad M \neq \mathcal{D}((p,q,g,x_1,x_2),(Y,W,U))] \\
&\leq \Pr[\boldsymbol{C} \text{ makes a query } (Y,W,U) \text{ for which } \boldsymbol{C}^*\text{'s response } M \text{ is such that} \\
&\qquad M = \perp \wedge \mathcal{D}((p,q,g,x_1,x_2),(Y,W,U)) \neq \perp] \\
&\leq \Pr[\boldsymbol{C} \text{ makes a query } (Y,W,U) \text{ for which } \boldsymbol{C}^*\text{'s response } M \text{ is such that} \\
&\qquad Y \not\equiv g^b \pmod{p} \wedge W \equiv Y^{\text{dlog}_g A} \pmod{p}] \\
&\leq \mathbf{Adv}^{\text{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(k)
\end{aligned}$$

The last inequality follows from the definition of dhk1-adversary $\boldsymbol{H}$. ▌

To prove that DEG is IND-CPA secure under the DDH assumption, we use the fact that if this assumption holds, then the ElGamal scheme EG is IND-CPA secure. Let $\boldsymbol{Y}$ be an ind-cpa-adversary attacking DEG. Consider the ind-cpa-adversary $\boldsymbol{Y}'$ attacking EG depicted in Figure 12. A random tape $x_1 \| R[\boldsymbol{Y}]$ for adversary $\boldsymbol{Y}'$ has two parts. The first part is a choice x_1 of an exponent in $\mathbb{Z}_q$ that $\boldsymbol{Y}'$ uses to compute the public key (p,q,g,X_1,X) and the challenge ciphertext C for $\boldsymbol{Y}$. The second part is a random tape for $\boldsymbol{Y}$. $\boldsymbol{Y}'$ runs $\boldsymbol{Y}$ and returns the output of the latter. Clearly, $\boldsymbol{Y}'$ runs in polynomial time and $\mathbf{Adv}^{\text{ind-cpa}}_{\text{EG},\boldsymbol{Y}'}(k) = \mathbf{Adv}^{\text{ind-cpa}}_{\text{DEG},\boldsymbol{Y}}(k)$, for every $k \in \mathbb{N}$. Since EG is IND-CPA secure, the ind-cpa-advantage of $\boldsymbol{Y}'$ is negligible, and hence it follows that the ind-cpa-advantage of $\boldsymbol{Y}$ is negligible. Thus DEG is IND-CPA secure.

The proof that the DHK0 assumption implies DEG is PA0 secure is analogous to the proof that the DHK1 assumption implies DEG is PA1 secure. The difference is that the given ciphertext creator $\boldsymbol{C}$ attacking DEG makes a single oracle query, and thus the dhk1-adversary $\boldsymbol{H}$ attacking prime-order-group generator G is a dhk0-adversary. The DHK0 assumption then implies the existence of a polynomial-time dhk0-extractor $\boldsymbol{H}^*$ for $\boldsymbol{H}$. The extractor $\boldsymbol{C}^*$ defined in Figure 12 is then a pa0-extractor. The proof that $\boldsymbol{C}^*$ is a successful polynomial-time pa0-extractor for $\boldsymbol{C}$ is exactly as before.

dhk1-adversary $\boldsymbol{H}(p, q, g, A; R[\boldsymbol{H}])$

Parse $R[\boldsymbol{H}]$ as $g_2 \| x_2 \| Z \| R[\boldsymbol{C}]$ where $g_2 \in G_q$, $x_2 \in \mathbb{Z}_q$, and $Z \in G_q$; $X \leftarrow A \cdot g_2^{x_2} \bmod p$

Run $\boldsymbol{C}$ on input (p, q, g_1, g_2, X, Z) and coins $R[\boldsymbol{C}]$ until it halts, replying to its oracle queries as follows:

– If $\boldsymbol{C}$ makes query (R_1, R_2, E, V) then

$W \leftarrow V \cdot R_2^{-x_2} \bmod p$; Make query (R_1, W) ; Let b denote the response

If $\big(R_1 \not\equiv g_1^b \pmod p$ or $R_2 \not\equiv g_2^b \pmod p$ or $V \not\equiv X^b \pmod p\big)$ then $M \leftarrow \perp$

Else $M \leftarrow E \cdot Z^{-b} \bmod p$ EndIf

Return M to $\boldsymbol{C}$ as the reply EndIf

Halt

---

pa1-extractor $\boldsymbol{C}^*(Q, \mathrm{St}[\boldsymbol{C}^*]; R[\boldsymbol{C}^*])$

If $\mathrm{St}[\boldsymbol{C}^*]$ is the initial state then

Parse $\mathrm{St}[\boldsymbol{C}^*]$ as $((p, q, g_1, g_2, X, Z), R[\boldsymbol{C}])$; Parse $R[\boldsymbol{C}^*]$ as $x_2 \| R[\boldsymbol{H}^*]$ where $x_2 \in \mathbb{Z}_q$

$A \leftarrow X \cdot g_2^{-x_2} \bmod p$; $\mathrm{St}[\boldsymbol{H}^*] \leftarrow ((p, q, g, A), g_2 \| x_2 \| Z \| R[\boldsymbol{C}])$

Else Parse $\mathrm{St}[\boldsymbol{C}^*]$ as $((p, q, g_1, g_2, X, Z), \mathrm{St}[\boldsymbol{H}^*], R[\boldsymbol{H}^*])$ EndIf

Parse Q as (R_1, R_2, E, V) ; $W \leftarrow V \cdot R_2^{-x_2} \bmod p$; $(b, \mathrm{St}[\boldsymbol{H}^*]) \leftarrow \boldsymbol{H}^*((R_1, W), \mathrm{St}[\boldsymbol{H}^*]; R[\boldsymbol{H}^*])$

If $\big(R_1 \not\equiv g_1^b \pmod p$ or $R_2 \not\equiv g_2^b \pmod p$ or $V \not\equiv X^b \pmod p\big)$ then $M \leftarrow \perp$

Else $M \leftarrow E \cdot Z^{-b} \bmod p$ EndIf

$\mathrm{St}[\boldsymbol{C}^*] \leftarrow ((p, q, g_1, g_2, X, Z), \mathrm{St}[\boldsymbol{H}^*], R[\boldsymbol{H}^*])$

Return $(M, \mathrm{St}[\boldsymbol{C}^*])$

Figure 13: dhk1-adversary $\boldsymbol{H}$ and pa1-extractor $\boldsymbol{C}^*$ for the proof of Theorem 5.4.

5.3 Proof of Theorem 5.4

CS-lite is known to be IND-CCA1 secure (and hence IND-CPA secure) under the DDH assumption (cf. [13]). Therefore, it is sufficient to prove that it is PA1 secure under the DHK1 assumption and PA0 secure under the DHK0 assumption.

We begin with the former. Let $\boldsymbol{C}$ be a polynomial-time ciphertext creator attacking CS-lite. We build a polynomial-time pa1-extractor $\boldsymbol{C}^*$ for it. First, we construct a polynomial-time dhk1-adversary $\boldsymbol{H}$ attacking prime-order-group generator G. By the DHK1 assumption, $\boldsymbol{H}$ has a polynomial-time dhk1-extractor $\boldsymbol{H}^*$. We then use $\boldsymbol{H}^*$ to build $\boldsymbol{C}^*$. Algorithms $\boldsymbol{H}$ and $\boldsymbol{C}^*$ are defined in Figure 13.

The random tape $g_2 \| x_2 \| Z \| R[\boldsymbol{C}]$ of $\boldsymbol{H}$ consists of a choice g_2 of an element in the group G_q, a choice x_2 of an exponent in $\mathbb{Z}_q$, a choice Z of an element in G_q, and a random tape $R[\boldsymbol{C}]$ for $\boldsymbol{C}$. The random tape $x_2 \| R[\boldsymbol{H}^*]$ of pa1-extractor $\boldsymbol{C}^*$ consists of a choice x_2 of an exponent in $\mathbb{Z}_q$ and a random tape for dhk1-extractor $\boldsymbol{H}^*$. $\boldsymbol{C}^*$ uses x_2 to compute value A and a random tape $g_2 \| x_2 \| Z \| R[\boldsymbol{C}]$ corresponding to $\boldsymbol{H}$, for $\boldsymbol{H}^*$. While extractor $\boldsymbol{C}^*$ gets input the random tape $R[\boldsymbol{C}]$ of $\boldsymbol{C}$, extractor $\boldsymbol{H}^*$ must be given input the random tape $R[\boldsymbol{H}] = g_2 \| x_2 \| Z \| R[\boldsymbol{C}]$ of $\boldsymbol{H}$. Clearly, $\boldsymbol{H}$ and $\boldsymbol{C}^*$ are polynomial time. We claim that $\boldsymbol{C}^*$ is a successful pa1-extractor for $\boldsymbol{C}$. To prove this, let $\boldsymbol{D}$ be a polynomial-time distinguisher for $\boldsymbol{C}$, and fix $k \in \mathbb{N}$. We state a claim, conclude the proof given this claim and Lemma 5.5, and then return to prove the claim.

Claim 5.7 Let DecOK denote the event that all $\boldsymbol{C}^*$'s answers to $\boldsymbol{C}$'s queries are correct in experiment $\mathbf{Exp}^{\text{pa1-x}}_{\text{CS-lite},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k)$. Then there exists a negligible function $\nu_{\boldsymbol{D}}$ such that $\Pr\left[\,\overline{\text{DecOK}}\,\right] \leq \mathbf{Adv}^{\text{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(k) + \nu_{\boldsymbol{D}}(k)$. ∎

Analogously to the proof of Theorem 5.3, Lemma 5.5 and Claim 5.7 imply that

$$\mathbf{Adv}^{\text{pa1}}_{\text{CS-lite},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) \;\leq\; \mathbf{Adv}^{\text{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(k) + \nu_{\boldsymbol{D}}(k)\,.$$

By the DHK1 assumption, the function $\mathbf{Adv}^{\text{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(\cdot)$ is negligible and hence for every polynomial-time distinguisher $\boldsymbol{D}$, the function $\mathbf{Adv}^{\text{pa1}}_{\text{CS-lite},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(\cdot)$ is negligible. Thus $\boldsymbol{C}^*$ is a successful pa1-extractor for $\boldsymbol{C}$, and CS-lite is PA1 secure. It remains to prove Claim 5.7.

Sketch of Proof of Claim 5.7: We call $(R_1, R_2, E, V) \in G_q^4$ a *valid ciphertext* with respect to public key (p, q, g_1, g_2, X, Z) if $\text{dlog}_{g_1} R_1 = \text{dlog}_{g_2} R_2$, and an *invalid ciphertext* otherwise. Cramer and Shoup [13] proved that the decryption algorithm of their IND-CCA2-secure scheme rejects all invalid ciphertexts generated by an adversary with all but negligible probability. By slightly modifying their proof, we can show that the decryption algorithm of CS-lite rejects all invalid ciphertexts generated by an adversary with all but negligible probability. Using this fact, we can prove that

$$\mathbf{Adv}^{\text{pa1}}_{\text{CS-lite},\boldsymbol{C},\boldsymbol{D},\boldsymbol{C}^*}(k) \;\leq\; \mathbf{Adv}^{\text{dhk1}}_{G,\boldsymbol{H},\boldsymbol{H}^*}(k) + \nu_{\boldsymbol{D}}(k)\,,$$

for a negligible function $\nu_{\boldsymbol{D}}$. Details are omitted. ∎

The proof that the DHK0 assumption implies CS-lite is PA0 secure is analogous to the proof that the DHK1 assumption implies CS-lite is PA1 secure. The difference is that the given ciphertext creator $\boldsymbol{C}$ attacking CS-lite makes a single oracle query, and thus the dhk1-adversary $\boldsymbol{H}$ attacking prime-order-group generator G is a dhk0-adversary. The DHK0 assumption then implies the existence of a polynomial-time dhk0-extractor $\boldsymbol{H}^*$ for $\boldsymbol{H}$. The extractor $\boldsymbol{C}^*$ defined in Figure 13 is then a pa0-extractor. The proof that $\boldsymbol{C}^*$ is a successful polynomial-time pa0-extractor for $\boldsymbol{C}$ is exactly as before.

References

[1] M. Backes, B. Pfitzmann and M. Waidner. A composable cryptographic library with nested operations. *Proceedings of the 10th Annual Conference on Computer and Communications Security*, ACM, 2003.

[2] B. Barak. How to go beyond the black-box simulation barrier. *Proceedings of the 42nd Symposium on Foundations of Computer Science*, IEEE, 2001.

[3] M. Bellare, A. Boldyreva and A. Palacio. An un-instantiable random oracle model scheme for a hybrid encryption problem. *Advances in Cryptology – EUROCRYPT '04*, Lecture Notes in Computer Science Vol. ?? , C. Cachin and J. Camenisch ed., Springer-Verlag, 2004.

[4] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. Relations among notions of security for public-key encryption schemes. *Advances in Cryptology – CRYPTO '98*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.

[5] M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. *Advances in Cryptology – CRYPTO '04*, Lecture Notes in Computer Science Vol. 3152 , M. Franklin ed., Springer-Verlag, 2004.

[6] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *Proceedings of the 1st Annual Conference on Computer and Communications Security*, ACM, 1993.

[7] M. Bellare and P. Rogaway. Optimal asymmetric encryption. *Advances in Cryptology – EUROCRYPT '94*, Lecture Notes in Computer Science Vol. 950, A. De Santis ed., Springer-Verlag, 1994.

[8] D. Boneh. Simplified OAEP for the RSA and Rabin functions. *Advances in Cryptology – CRYPTO '01*, Lecture Notes in Computer Science Vol. 2139, J. Kilian ed., Springer-Verlag, 2001.

[9] M. Blum, P. Feldman and S. Micali. Non-interactive zero-knowledge and its applications. *Proceedings of the 20th Annual Symposium on the Theory of Computing*, ACM, 1988.

[10] M. Blum, P. Feldman and S. Micali. Proving security against chosen ciphertext attacks. *Advances in Cryptology – CRYPTO '88*, Lecture Notes in Computer Science Vol. 403, S. Goldwasser ed., Springer-Verlag, 1988.

[11] R. Canetti, O. Goldreich and S. Halevi. The random oracle methodology, revisited. *Proceedings of the 30th Annual Symposium on the Theory of Computing*, ACM, 1998.

[12] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. *Advances in Cryptology – CRYPTO '98*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998.

[13] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, Vol. 33, No. 1, 2003, pp. 167–226.

[14] I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. *Advances in Cryptology – CRYPTO '91*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.

[15] A. De Santis and G. Persiano. Zero-knowledge proofs of knowledge without interaction. *Proceedings of the 33rd Symposium on Foundations of Computer Science*, IEEE, 1992.

[16] D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, Vol. 29, 1983, pp. 198–208.

[17] D. Dolev, C. Dwork, and M. Naor. Non-Malleable cryptography. *SIAM Journal on Computing*, Vol. 30, No. 2, 2000, pp. 391–437.

[18] E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern. RSA-OAEP is secure under the RSA assumption. *Advances in Cryptology – CRYPTO '01*, Lecture Notes in Computer Science Vol. 2139, J. Kilian ed., Springer-Verlag, 2001.

[19] O. Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, Vol. 6, No. 1, 1993, pp. 21–53.

[20] S. Goldwasser and Y. Taumann. On the (in)security of the Fiat-Shamir paradigm. FOCS 03.

[21] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. IACR Cryptology ePrint Archive, Report 1999/009, March 1999. Available at `http://eprint.iacr.org/1999/009/`. [Revised version of [22].]

[22] S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. *Advances in Cryptology – CRYPTO '98*, Lecture Notes in Computer Science Vol. 1462, H. Krawczyk ed., Springer-Verlag, 1998. [Preliminary version of [21], superceded by the latter.]

[23] J. Herzog, M. Liskov and S. Micali. Plaintext awareness via key registration. *Advances in Cryptology – CRYPTO '03*, Lecture Notes in Computer Science Vol. 2729, D. Boneh ed., Springer-Verlag, 2003.

[24] S. Goldwasser and S. Micali. Probabilistic Encryption. *Journal of Computer and System Science*, Vol. 28, 1984, pp. 270–299.

[25] S. Micali, C. Rackoff, and B. Sloan. The notion of security for probabilistic cryptosystems. *SIAM Journal on Computing*, Vol. 17, No. 2, 1988, pp. 412–426.

[26] M. Naor. Cryptographic assumptions and challenges. *Advances in Cryptology – CRYPTO '03*, Lecture Notes in Computer Science Vol. 2729, D. Boneh ed., Springer-Verlag, 2003.

[27] M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. STOC 90.

[28] J. B. Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. *Advances in Cryptology – CRYPTO '02*, Lecture Notes in Computer Science Vol. 2442, M. Yung ed., Springer-Verlag, 2002.

[29] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. *Advances in Cryptology – CRYPTO '91*, Lecture Notes in Computer Science Vol. 576, J. Feigenbaum ed., Springer-Verlag, 1991.

[30] V. Shoup. OAEP reconsidered. *Journal of Cryptology* Vol. 15, No. 4, 2002, pp. 223–249.

A Damgård's arguments about the security of DEG

We first review Damgård's security notions and then his proof.

RPR security. Let $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathrm{MsgSp})$ be an encryption scheme. Damgård [14] considers security against recovery of a random plaintext under a type 1 chosen-ciphertext attack. Namely, let us say that $\mathcal{AE}$ is RPR-CCA1 secure if for every polynomial time $\boldsymbol{R}$, the probability that the following experiment returns 1 is negligible as a function of k:

$(pk, sk) \xleftarrow{\$} \mathcal{K}(1^k)$; $\mathrm{St} \xleftarrow{\$} \boldsymbol{R}^{\mathcal{D}_{sk}(\cdot)}(\mathsf{find}, pk)$; $M \xleftarrow{\$} \mathrm{MsgSp}(pk)$; $C \xleftarrow{\$} \mathcal{E}(pk, M)$; $M' \leftarrow \boldsymbol{R}(\mathsf{guess}, C, \mathrm{St})$
If $M = M'$ then return 1 else return 0

One can show that IND-CCA1 $\rightarrow$ RPR-CCA1 and RPR-CCA1 $\not\rightarrow$ IND-CCA1, meaning this notion of security is weaker than IND-CCA1 security. One can define RPR-CPA security too, just by not giving $\boldsymbol{R}$ the decryption oracle in the first stage above.

Claim and proof approach. Damgård [14, Theorem 2] claims that DEG is RPR-CCA1 secure assuming DHK0 and the security of the ElGamal encryption scheme under RPR-CPA. He first shows that if ElGamal is RPR-CPA then so is DEG [14, Lemma 1]. His proof of his Theorem 2 [14, Page 453] claims to turn a given RPR-CCA1 adversary into an RPR-CPA adversary. Applying his Lemma 1, he can conclude. The issue is how an RPR-CCA1 adversary $\boldsymbol{R}$ is turned into an RPR-CPA one. Quoting from the proof of [14, Page 453], with some minor changes for consistency with our notation:

> Let $C_1, C_2, \ldots$ be the sequence of ciphertexts whose decryptions $\boldsymbol{R}$ requests from its oracle. Let $\boldsymbol{H}_i$ be the algorithm that simulates $\boldsymbol{R}$ until the output of C_i and then stops. We can now show by induction that for all i, $\boldsymbol{H}_i$ can be simulated without access to a decryption oracle. $\boldsymbol{H}_1$ is clear. To do $\boldsymbol{H}_{i+1}$, observe that by induction, $\boldsymbol{H}_i$ can be simulated without the oracle. Then the DHK0 assumption guarantees us the existence of an algorithm $\boldsymbol{H}_i^*$ that outputs y where $C_i = (Y, W, U)$ and $Y = g^y$, whenever C_i produces a non-null output from the decryption. Knowledge of y suffices to decrypt C_i, and therefore we can simulate also the last steps of $\boldsymbol{R}_{i+1}$. From $\boldsymbol{R}$ we can therefore *build an algorithm that breaks the system* under an RPR-CPA attack, and we are done by Lemma 1.

The problem is the emphasized text at the end of the quoted proof above. An *algorithm* is by definition a *finite* object. We could view it as a program, or, more formally, as a Turing machine, but it must have a finite description of size independent of the size of the input. However, the algorithm resulting from the above proof contains descriptions of the extractors $\boldsymbol{H}_1^*, \boldsymbol{H}_2^*, \ldots$ which it must run as subroutines. We claim this list is infinite, so that the constructed "algorithm" is actually an object having an infinite description, and not an algorithm at all.

Why is the list of extractors infinite? The list is finite for any given value of the security parameter. But suppose $\boldsymbol{R}$ makes $q(k) = k$ oracle queries. The constructed algorithm must work for any value of k. So it must include the list of extractors corresponding to all values of k, and this list is unbounded.

The easiest fix to the above is to use the DHK1 assumption instead. This guarantees a single extractor that can interactively take inputs and extract the appropriate quantities from them. In that case, Damgård's proof goes through to show that DEG is RPR-CCA1 secure assuming DHK1 and the

RPR-CPA security of ElGamal. (Note we show something somewhat stronger, namely IND-CCA1 security, and we also show PA0, PA1.)

We remark that a strategy similar to Damgård's is used by [4] in their proof that PA-BDPR+IND-CPA implies IND-CCA2 in the RO model. They can avoid having their algorithm remember infinitely many extractors because in their definition of PA-BDPR the extractor does not depend on the adversary.