Relations among Notions of Complete Non-Malleability: Indistinguishability Characterisation and Efficient Construction without Random Oracles

M. Barbosa[1] and P. Farshim[2]

[1] CCTC/Departamento de Informática, Universidade do Minho,
Campus de Gualtar, 4710-057 Braga, Portugal.
mbb@di.uminho.pt
[2] Information Security Group, Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, United Kingdom.
Pooya.Farshim@rhul.ac.uk

Abstract. We study relations among various notions of complete non-malleability, where an adversary can tamper with both ciphertexts and public-keys, and ciphertext indistinguishability. We follow the pattern of relations previously established for standard non-malleability. To this end, we propose a more convenient and conceptually simpler indistinguishability-based security model to analyse completely non-malleable schemes. Our model is based on strong decryption oracles, which provide decryptions under arbitrarily chosen public keys. We give the first precise definition of a strong decryption oracle, pointing out the subtleties in different approaches that can be taken. We construct the first *efficient* scheme, which is fully secure against strong chosen-ciphertext attacks, and therefore completely non-malleable, without random oracles.

Keywords. Complete Non-Malleability. Strong Chosen-Ciphertext Attacks. Public-Key Encryption. Provable Security.

1 Introduction

BACKGROUND. The security of public-key encryption schemes has been formalised according to various goals and attack models. Extensive work has been done in establishing relations between these security notions, and converging towards a core set of standard security definitions. Well-studied goals include semantic security, indistinguishability, and non-malleability; whereas chosen-plaintext and (adaptive) chosen-ciphertext are the most common attack scenarios considered in literature.

An important criterion for selecting security models is the guarantee of *necessary security* for a class of applications with practical relevance. Conversely, it is also expected that one can select a security model that is only as strict as required by a specific application. Otherwise, one might rule out valid solutions without justification, possibly sacrificing other important factors such as set-up assumptions, computational cost or communications bandwidth. Another important criterion is the conceptual simplicity and ease of use of a model.

Indistinguishability of ciphertexts is the most widely used notion of security for public-key encryption schemes. This notion was proposed by Goldwasser and Micali [GM84] as a convenient formalisation of the more intuitive notion of semantic security. Other notions of security have been proposed in different contexts. Of particular interest to this work is non-malleability, initially proposed by Dolev, Dwork, and Naor [DDN00]. Roughly speaking, an encryption scheme is non-malleable if giving an encryption of a message to an adversary does not increase its chances of producing an encryption of a related message (under a given public key). This is formalised by requiring the existence of a simulator that performs as well as the adversary but without seeing the original encryption.

The relations between different notions of security for public-key encryption schemes were examined in a systematic way by Bellare et al. [BDPR98]. There, the authors compare indistinguishability of ciphertexts and non-malleability under chosen-plaintext and chosen-ciphertext attacks. In doing so, they formalise a comparison-based definition of non-malleability and establish important results based on this: non-malleability implies indistinguishability for an equivalent attack model, there is an equivalence between these notions for adaptive chosen-ciphertext attacks, and there are separations between the two notions for intermediate attack models.

Bellare and Sahai [BS06] established a cycle of equivalence between three definitions of non-malleability: a simulation-based definition similar to that of Dolev, Dwork and Naor, a comparison-based definition as introduced

in [BDPR98], and a new definition called indistinguishability of ciphertexts under parallel chosen-ciphertext attacks. These equivalence relations essentially establish that the three definitions are alternative formulations of the same notion. Pass, Shelat, and Vaikuntanathan [PSV07] revisit this equivalence result, and clarify several technical aspects in the known equivalence proofs. They consider the important question of composability of definitions, and establish a separation between the simulation-based and comparison-based non-malleability definitions, showing that the former is strictly stronger for *general* schemes.

Besides being theoretically interesting, the above results are also relevant in practice. They permit designers of encryption schemes to base their analysis on the simpler and better understood IND-CCA2 security model. This facilitates the presentation of conceptually simpler proofs, which are less prone to errors, as well as the direct application of a well-known set of proof techniques.

COMPLETE NON-MALLEABILITY. Fischlin [Fis05] introduces a stronger notion of non-malleability, known as *complete*, which requires attackers to have negligible advantage, even if they are allowed to transform the public key under which the related message is encrypted. Put differently, the goal of an adversary is to construct a related ciphertext under a new public key pair, for which the attacker might not even know a valid secret key.

Fischlin shows that well-known encryption schemes such as Cramer-Shoup [CS98] and RSA-OAEP [FOPS04] do *not* achieve even the weakest form of complete non-malleability. Furthermore, he proves a negative results with respect to the existence of completely non-malleable schemes for general relations: there is a large class of relations for which completely non-malleable schemes do not exist with respect to black-box simulators. On the other hand, Fischlin establishes a positive result for a modified version of RSA-OAEP, with respect to a restricted class of adversaries, in the random oracle model.

Ventre and Visconti [VV08] later propose a comparison-based definition of this security notion, which is more in line with the well-studied definitions proposed by Bellare et al. [BDPR98,BS06]. For chosen-plaintext attacks the authors prove that (a restricted version of) their definition is equivalent to that of Fischlin. They also establish equivalence for chosen-ciphertext attacks, for a well-defined class of relations that do not depend on the challenge public key (known as lacking relations). The authors also provide additional feasibility results by proposing two constructions of completely non-malleable schemes, one in the common reference string model using non-interactive zero-knowledge proofs, and another using interactive encryption schemes. Therefore, the only previously known completely non-malleable (and non-interactive) scheme in the standard model, is quite inefficient as it relies on generic zero-knowledge techniques.

MOTIVATION. The initial motivation for complete non-malleability resided on constructing *non-malleable commitment* schemes. A commitment scheme can be constructed from an encryption scheme in the following way. To commit to a message, one generates a key pair and encrypts the message under the generated public key. The resulting public key/ciphertext pair forms the commitment. To de-commit, one reveals a valid secret key or the message/randomness pair used in encryption. In this setting, it is clearly desirable that the encryption scheme should be completely non-malleable in order to guarantee non-malleability of the associated commitment scheme.

Furthermore, new notions of security of high practical relevance have been emerging in the literature that closely relate to different flavours of complete non-malleability. The pattern connecting these notions is that adversaries are allowed to tamper with the keys, under which they are challenged, in order to gain extra advantage. *Robust encryption* [ABN10] is one such notion, and it is pitched at applications where ciphertext anonymity is relevant. This notion requires it to be infeasible to construct a ciphertext which is valid under two distinct public keys. Another such notion is security under *related-key attacks* [BK03], where cipher operations can be executed over perturbed versions of the challenge secret key. This model is of particular relevance in the symmetric encryption setting. Also worth mentioning are concrete attacks on key-agreement protocols and public-key signature schemes, where attackers are able to introduce public keys of their choice in the protocol execution [Fis05].

The relations between these new notions of security are understudied and constitute a novel challenge in theoretical cryptography. A deeper understanding of the relations between these notions of security should permit identifying a core set of security models that facilitate the design and analysis of strongly secure schemes with

practical relevance. The main motivation of this work is, therefore, to take an important step in this direction. We aim to expand the current understanding of complete non-malleability, by establishing relations among notions of complete non-malleability and ciphertext indistinguishability that are akin to those already known for standard non-malleability. To this end, we introduce a new indistinguishability based notion, and demonstrate its applicability by constructing an efficient and completely non-malleable scheme.

STRONG CHOSEN-CIPHERTEXT ATTACKS. Our search for a suitable indistinguishability-based definition of complete non-malleability resulted in a natural extension of the standard IND-CCA2 security model, in which the adversary can get decryptions of ciphertexts under arbitrary public keys of its choice. We call this a *strong chosen-ciphertext attack* scenario, and say that the adversary is given access to a *strong decryption oracle*. This, in turn, brings together two fields which previously remained unrelated in provable security, namely complete non-malleability and certificateless cryptography [ARP03,DLP08]. Indeed, strong chosen-ciphertext attacks can be seen to model multi-user scenarios where public keys might not be authenticated, and were initially proposed as a natural attack model for certificateless schemes that aimed to do away with public-key certificates.

The question of whether the weakness captured by such a strong model should be seen as a real vulnerability of public-key encryption schemes has caused some discussion [DLP08]. Arguments against this approach are based on the fact that such an attack model is not realistic, since it is highly unlikely that the adversary is able to get such assistance in a practical scenario. Another way to put this objection is that security models should be defined through experiments that are guaranteed to execute in polynomial time: providing decryptions under unknown secret keys assists the adversary through a super-polynomial time oracle.

The results we present in this paper show that the strength of the complete non-malleability notion is comparable to that of the strong chosen-ciphertext attack scenario. This connection allows us to take a more constructive view of strong decryption oracles, and argue that they can indeed be useful to analyse the security of practical schemes. To support this view, we show that *indistinguishability under strong CCA attacks is a convenient formalisation to establish that a scheme is completely non-malleable*. Furthermore, by proposing a concrete scheme, we also show that both notions are realisable without random oracles.

Finally, we note that strong decryption oracles are closely related to the recently proposed paradigm of adaptive one-way functions [PPV08], which can be used to construct a number of cryptographic protocols that previously remained open in the literature. Indeed, the assumptions that underlie the proposed constructions of adaptive one-way functions rely on similar "magic" oracles. It would be interesting to investigate whether the techniques that we use can be useful in constructing adaptive one-way functions based on standard assumptions. Conversely, the public-key encryption scheme given in [PPV08] seems to achieve strong chosen-ciphertext security. The relationship between adaptive one-way functions and strong security models are left for future work.

CONTRIBUTIONS. The first contribution of our paper is a general definition of a strong decryption oracle, which unifies previous definitional approaches. Our definition is flexible and expressive in the sense that it allows identifying the exact power of the decryption oracle that is provided to an adversary in security analysis. We also show that variants of the strong decryption oracle definition map to interesting properties of encryption schemes. We establish a connection with the validity checks that an encryption scheme performs (message validity, ciphertext validity, public key validity, etc.). More precisely, *we identify a simple and very convenient definition of the strong decryption oracle, which can be used to analyse schemes that incorporate a well-defined and natural set of validity checks*. For schemes that fail to perform these checks, care must be taken to identify the exact strength of the strong decryption oracle under which the scheme can be proven secure.

We then extend the standard indistinguishability and non-malleability models using strong decryption oracles, and examine the relations between the resulting notions. Our approach is consistent with that proposed by Bellare et al. [BS06,BDPR98], which allows us to naturally describe the relation between these stronger models and the more established ones. We also identify the relation between the strong chosen-ciphertext models we propose and the existing notions of complete non-malleability. To the best of our knowledge, this relation was not previously known. It permits fully characterising how these independently proposed models relate to the more standard definitions

of non-malleability. The relation we establish between strong decryption oracles and complete non-malleability *provides the first convincing argument that the strong CCA models are useful in analysing the security of practical encryption schemes*.

Finally, we propose a concrete scheme that *efficiently achieves strong chosen-ciphertext security* based on the decisional bilinear Diffie-Hellman assumption. The scheme is secure under a very general definition of the strong decryption oracle, which is made possible by the insights regarding validity checks we described above. The scheme is derived from Waters' identity-based encryption scheme [Wat05] using techniques previously employed in constructing certificateless public-key encryption schemes [DLP08]. Our equivalence result also establishes our scheme as the first efficient completely non-malleable scheme without random oracles. We stress that our scheme is based on a standard and well-known problem and does *not* rely on interactive assumptions or "magic" oracles.

ORGANISATION. In the next section we fix notation by defining public-key encryption schemes and various algorithms associated to them. In Section 3 we discuss different approaches in defining strong decryption oracles and propose a new generic definition. In Section 4 we look at indistinguishability and non-malleability security models for encryption schemes where adversaries have access to strong decryption oracles. We establish relations between these models and also to models existing literature. Finally, in the last section we give the first practical completely non-malleable scheme without random oracles.

2 Preliminaries

NOTATION. We write $x \leftarrow y$ for assigning value y to variable x. We write $x \leftarrow_{\$} X$ for sampling x from set X uniformly at random. If X is empty, we set $x \leftarrow \perp$, where $\perp \notin \{0,1\}^\star$ is a special failure symbol. If A is a probabilistic algorithm we write $x \leftarrow_{\$} A(I_1, I_2, \ldots)$ for the action of running A on inputs $I_1, I_2, \ldots$ with random coin chosen uniformly at random, and assigning the result to x. Sometimes we run A on specific coins r and write $x \leftarrow A(I_1, I_2, \ldots; r)$. We denote boolean values, namely the output of checking whether a relation holds, by T (true) and F (false). For a space $\mathsf{Sp} \subseteq \{0,1\}^\star$, we identify Sp with its characteristic function. In other words, $\mathsf{Sp}(s) = \mathsf{T}$ if and only if $s \in \mathsf{Sp}$. The function $\mathsf{Sp}(\cdot)$ always exists, although it may not be computable in polynomial time. We say s is valid with respect to Sp if and only if $\mathsf{Sp}(s) = \mathsf{T}$. When this is clear from the context, we also use Sp for sampling uniformly from Sp. Unless stated otherwise, the range of a variable s is assumed to be $\{0,1\}^\star$. The symbol : is used for appending an element to a list. We indicate vectors using bold font.

GAMES. We will be using the code-based game-playing language [BR06]. Each game has an **Initialize** and a **Finalize** procedure. It also has specifications of procedures to respond to an adversary's various oracle queries. A game Game is run with an adversary $\mathcal{A}$ as follows. First **Initialize** runs and its outputs are passed to $\mathcal{A}$. Then $\mathcal{A}$ runs and its oracle queries are answered by the procedures of Game. These procedures return $\perp$ if queried on $\perp$. When $\mathcal{A}$ terminates, its output is passed to **Finalize** which returns the outcome of the game y. This interaction is written as $\mathsf{Game}^{\mathcal{A}} \Rightarrow y$. In each game, we restrict attention to *legitimate* adversaries. Legitimacy is defined specifically for each game.

PUBLIC-KEY ENCRYPTION. We adopt the standard multi-user syntax with the extra Setup algorithm [BBM00], which we believe is the most natural one for security models involving multiple public keys. A public-key encryption scheme $\Pi = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{MsgSp}, \mathsf{Enc}, \mathsf{Dec})$ is specified by five polynomial-time algorithms (in the length of their inputs) as follows. Setup is the probabilistic setup algorithm which takes as input the security parameter and returns the common parameters I (we fix the security parameter implicitly, as we will be dealing with concrete security). Although all algorithms are parameterised by I, we often omit I as an explicit input for readability. $\mathsf{Gen}(\mathsf{I})$ is the probabilistic key-generation algorithm. On input common parameters I, this algorithm returns a secret key SK and a matching public key PK. Algorithm $\mathsf{MsgSp}(\mathsf{m}, \mathsf{PK})$ is a deterministic message space recognition algorithm. On input m and PK this algorithm returns T or F. $\mathsf{Enc}(\mathsf{m}, \mathsf{PK}; r)$ is the probabilistic encryption algorithm. On input a message m, a public key PK, and possibly some random coins r, this algorithm outputs a ciphertext c or a special

failure symbol $\perp$. Finally, $\mathsf{Dec}(\mathsf{c}, \mathsf{SK}, \mathsf{PK})$ is the deterministic decryption algorithm. On input of a ciphertext c and keys SK and PK, it outputs a message m or a special failure symbol $\perp$. The correctness of a public-key encryption scheme requires that for any $\mathsf{I} \leftarrow_{\$} \mathsf{Setup}()$, any $(\mathsf{SK}, \mathsf{PK}) \leftarrow_{\$} \mathsf{Gen}()$, all $\mathsf{m} \in \mathsf{MsgSp}(\mathsf{PK})$, and any random coins r we have $\mathsf{Dec}(\mathsf{Enc}(\mathsf{m}, \mathsf{PK}; r), \mathsf{SK}, \mathsf{PK}) = \mathsf{m}$.

REMARK. We note that the multi-user syntax permits capturing in a single framework schemes that execute in the plain model, in which case the global parameters are empty, as well as those which execute in the CRS model. The relations that we establish between different models hold in both cases.

VALIDITY CHECKING ALGORITHMS. The following spaces (and associated functions) will be used throughout the paper. All of these spaces are parameterised by I and are subsets of $\{0,1\}^\star$.

$$\mathsf{MsgSp}(\mathsf{PK}) := \{\mathsf{m} : \mathsf{MsgSp}(\mathsf{m}, \mathsf{PK})\}$$
$$\mathsf{KeySp} := \{(\mathsf{SK}, \mathsf{PK}) : \exists r\ (\mathsf{SK}, \mathsf{PK}) = \mathsf{Gen}(r)\}$$
$$\mathsf{PKSp} := \{\mathsf{PK} : \exists r, \mathsf{SK}\ (\mathsf{SK}, \mathsf{PK}) = \mathsf{Gen}(r)\}$$
$$\mathsf{SKSp} := \{\mathsf{SK} : \exists r, \mathsf{PK}\ (\mathsf{SK}, \mathsf{PK}) = \mathsf{Gen}(r)\}$$

VALIDITY ASSUMPTIONS. We assume throughout the paper that the encryption and decryption algorithms check if $\mathsf{m} \in \mathsf{MsgSp}(\mathsf{PK})$ and return $\perp$ if it does not hold. Often the algorithm MsgSp does not depend on PK in the sense that for any $\mathsf{PK}, \mathsf{PK}' \in \mathsf{PKSp}$ and any $\mathsf{m} \in \{0,1\}^\star$ we have $\mathsf{MsgSp}(\mathsf{m}, \mathsf{PK}) = \mathsf{MsgSp}(\mathsf{m}, \mathsf{PK}')$. For general schemes, case one can consider the infinite message space $\mathsf{MsgSp}(\mathsf{PK}) = \{0,1\}^\star$. However, given that in this paper we will often consider the set of all valid messages and sample from it, we restrict our attention to schemes with finite message spaces. As pointed out by Pass et al. [PSV07], this means that to avoid degenerate cases we must also restrict our attention to schemes for which all the elements in the range of decryption can be efficiently encrypted, including the special failure symbol $\perp$. A distribution M on messages is *valid* with respect to a public key PK if it is computable in polynomial time and its support contains strings of equal length which lie in $\mathsf{MsgSp}(\mathsf{PK})$. We also assume that key-pair validity KeySp is efficiently implementable and require that decryption returns $\perp$ if this check fails on the keys passed to it (note that this can easily be achieved for general public key encryption schemes, by including the input randomness to Gen in SK). We also assume various algorithms check for structural properties such as correct encoding, membership in a group, etc.

3 Defining strong decryption oracles

The idea behind a strong chosen-ciphertext attack is to give the adversary access to an oracle that decrypts ciphertexts of the adversary's choice with respect to arbitrary public keys. There are a number technicalities involved in defining such an oracle precisely, which we now discuss.

procedure $\mathbf{SDecrypt}_{\mathsf{U},\mathsf{V}}(\mathsf{c}, \mathsf{PK}, \mathsf{R})$:
$\mathsf{WitSp} \leftarrow \{(\mathsf{m}, r) : \mathsf{V}(\mathsf{c}, \mathsf{PK}, \mathsf{m}, r, \mathsf{st}[\mathsf{V}])\}$
$(\mathsf{m}, r) \leftarrow_{\$} \{(\mathsf{m}, r) \in \mathsf{WitSp} : \mathsf{R}(\mathsf{m})\}$
$\mathsf{st}[\mathsf{V}] \leftarrow \mathsf{U}(\mathsf{c}, \mathsf{PK}, \mathsf{R}, \mathsf{m}, r, \mathsf{st}[\mathsf{V}])$
Return m

Fig. 1: Generic definition of a strong decryption oracle. In the first step the search is performed over sufficiently long bit strings and, for messages, it also includes the special symbol $\perp$. The state $\mathsf{st}[\mathsf{V}]$ is initialised to some value st_0.

We will base our presentation on the generic definition of a strong decryption oracle presented in Figure 1, which we thoroughly explain and justify in the discussion that follows. The oracle proceeds in three steps. The first step models the general procedure of constructing a set of candidate (valid) decryption results. The second step consists of choosing one of these candidate solutions to return to the adversary. The final step updates the state of the oracle, if it keeps one.

More precisely, in the first step, the oracle constructs a set of possible decryption results WitSp using a polynomial-time validity relation V[3]. Note that the search for messages includes the special failure symbol $\perp$. This permits making the subtle distinction of returning $\perp$ when a candidate decryption result has not been found[4], or when it has been established that the oracle may return $\perp$ when queried on a given $(\mathsf{c}, \mathsf{PK})$ pair. In the second step, it selects the message to return from WitSp. To make sure the security model is not restricting the adversary by choosing the decryption result in a particular way, we allow the adversary to provide a polynomial-time relation R to characterise a set of messages of interest to her. The oracle then samples a message at random from this set and returns it to the adversary. In the third and final step, the oracle updates any state it may have stored from previous queries. We require that the update procedure to be polynomial in the size of its inputs, excluding the state[5].

Although we have constrained the algorithms in our definition (i.e. V, R and U) to be polynomial-time, the calculations carried out in the first two steps may not be computable in polynomial time and may require an exponential number of executions of these algorithms. Nevertheless, we emphasise that the search space must be finite. This is guaranteed by the assumption that the message space of the encryption scheme is finite, and by the fact that the algorithms associated with the scheme run in polynomial time in their inputs.

The motivation for having such a general definition is that the notion of *the message encapsulated by the ciphertext* can be defined in various ways. For concreteness, let us fix U so that $\mathsf{st}[\mathsf{V}]$ is empty throughout the game execution, and look at two alternative definitions of V. These derive from two interpretations as to which message(s) might be encapsulated in a public key/ciphertext pair: they can be seen as alternative witnesses to the validity of the public key/ciphertext pair. Concretely one can define validity via the encryption operation, in which case a message/randomness pair is the witness or via the decryption algorithm, in which case the natural witness is a message/secret key pair[6]:

$$\mathsf{V}(\mathsf{c}, \mathsf{PK}, \mathsf{m}, r) := \mathsf{c} \stackrel{?}{=} \mathsf{Enc}(\mathsf{m}, \mathsf{PK}; r) \tag{1}$$

$$\mathsf{V}'(\mathsf{c}, \mathsf{PK}, \mathsf{m}, r) := (\mathsf{SK}, \mathsf{PK}) \stackrel{?}{=} \mathsf{Gen}(r) \wedge \mathsf{m} \stackrel{?}{=} \mathsf{Dec}(\mathsf{c}, \mathsf{SK}, \mathsf{PK}). \tag{2}$$

The first observation to make on these validity criteria is that neither of them guarantees that if a message is found to be a valid decryption result, it will be unique. This is because the correctness restriction only guarantees unique decryptability for correctly constructed $(\mathsf{c}, \mathsf{PK})$ pairs: it says nothing about the result of decryption when an invalid public key and/or an invalid ciphertext are provided as inputs. In particular, the validity criterion in Equation 1 could accept multiple messages as valid, when run on an invalid public key. Ambiguity can also occur for the validity criterion in Equation 2, when multiple valid secret keys correspond to the queried public key, and decrypt an invalid ciphertext inconsistently. This discussion justifies the need for the second step in the definition we propose: there could be many valid decryption results to choose from, and it is left to the adversary to control how this is done. In the simplest scenario, where there is only one candidate decryption result, one can assume without loss of generality that the adversary will choose to retrieve that result by passing in the trivial relation T.

The need for the first step of the definition is justified by observing that the two witness sets associated with the above validity algorithms do not always coincide. To see this, consider an encryption scheme where decryption does not necessarily fail when run on a ciphertext that falls outside the range of the encryption algorithm. Then the first witness set will be empty whereas the second may not be. A concrete example is the Cramer-Shoup [CS98] encryption scheme. For other schemes, such as RSA-OAEP [FOPS04], it may happen that the encryption algorithm produces apparently valid ciphertexts for invalid public keys. When this is the case, the first witness set may not be empty, whereas the second one will surely contain no messages, given that no valid secret key exists.

We note that the above issues do not arise in the standard definition of a decryption oracle, in which decryption is always carried out with a fixed secret key. In other words, the decryption oracle is stateful. To allow capturing

[3] This constitutes an NP-relation for the language of valid decryption results.

[4] Recall that we assume that sampling from an empty set returns $\perp$.

[5] Discarding the state size ensures that the run-time of this procedure does not increase exponentially with queries.

[6] Note that we have assumed Dec always performs the key-pair validity check, and so this is redundant in V'. We include it for the sake of clarity: for schemes which do not perform the key-pair validity check, this issue must be considered.

this sort of behaviour in strong decryption oracles, we add the last step to the oracle definition. This manages the decryption oracle state, and ensures that the validity checking algorithm can access it in each query.

SPECIFIC DEFINITIONS. Previous attempts to define strong decryption oracles have been introduced for certificate-less public-key encryption, where public keys are not authenticated [ARP03,DLP08]. These definitions implicitly adopt validity criteria which are adequate only for the concrete schemes discussed in the referred works.

In the definition proposed in [ARP03] the authors simply describe the oracle as providing "correct decryptions" even though the secret key could be unknown. A close analysis of the presentation in this work indicates that "correct decryption" is defined through a search for a message/randomness pair in the domain of the encryption, similarly to the first validity criterion presented above. However, the unique decryptability issue is implicit in the definition, since the concrete scheme the authors consider ensures that the encryption algorithm fails when queried with an invalid public key. Extending this definition to encryption schemes in general results in the following validity criterion, which explicitly checks for public key validity:

$$\mathsf{V}_{\mathsf{PK}}(\mathsf{c}, \mathsf{PK}, \mathsf{m}, r||r') := \mathsf{c} \stackrel{?}{=} \mathsf{Enc}(\mathsf{m}, \mathsf{PK}; r) \wedge (\star, \mathsf{PK}) \stackrel{?}{=} \mathsf{Gen}(r').$$

Note that this is equivalent to the validity relation in Equation 1 for schemes which check for public key validity in the encryption algorithm. Alternatively, a solution adopted in literature [Fis05] is to restrict the class of adversaries to those which query only valid public keys. In our view, such a restriction on the adversary's behaviour is unjustified, and we will look for alternatives which guarantee stronger security.

In a more recent work [DLP08], the strong decryption oracle is described as constructing a private key that corresponds to the queried valid public key, and then using that key to decrypt the ciphertext. The oracle then stores the extracted secret key to be reused in subsequent queries under the same public key. This definition is more in line with the intuition that a decryption oracle should reflect the behaviour of the decryption algorithm, and it is also consistent with the stateful operation of the standard decryption oracle. We can capture this definition through the algorithms presented in Figure 2. Note that, for those schemes in which there is a unique valid private key per public key or for those schemes where all valid secret keys behave consistently for all possible, even invalid, ciphertexts, the oracle resulting from these algorithms will be identical to the one using the criterion in Equation 2.

procedure $\mathsf{V}(\mathsf{c}, \mathsf{PK}, \mathsf{m}, r, \mathsf{st}[\mathsf{V}])$:	**procedure** $\mathsf{U}(\mathsf{c}, \mathsf{PK}, \mathsf{R}, \mathsf{m}, r, \mathsf{st}[\mathsf{V}])$:
$(\mathsf{SK}', \mathsf{PK}') \leftarrow \mathsf{Gen}(r)$	$(\mathsf{SK}', \mathsf{PK}') \leftarrow \mathsf{Gen}(r)$
If $\mathsf{PK}' \neq \mathsf{PK} \vee ((\mathsf{SK}, \mathsf{PK}) \in \mathsf{st}[\mathsf{V}] \wedge \mathsf{SK}' \neq \mathsf{SK})$ Return F	If $\mathsf{PK}' \neq \mathsf{PK} \vee (\mathsf{SK}, \mathsf{PK}) \in \mathsf{st}[\mathsf{V}]$, Return $\mathsf{st}[\mathsf{V}]$
If $\mathsf{m} = \mathsf{Dec}(\mathsf{c}, \mathsf{SK}', \mathsf{PK}')$ Return T	$\mathsf{st}[\mathsf{V}] \leftarrow (\mathsf{SK}', \mathsf{PK}') : \mathsf{st}[\mathsf{V}]$
Return F	Return $\mathsf{st}[\mathsf{V}]$

Fig. 2: Update and validity algorithms for a stateful strong decryption oracle with initial state $\mathsf{st}_0 = (\mathsf{SK}^\star, \mathsf{PK}^\star)$.

The previous discussion indicates that different definitions of a strong decryption oracle can be seen as natural for particular classes of schemes. However, we can also consider other approaches, which are not so easy to characterise. For example, a straightforward fix to the ambiguity problem described above is to have the oracle simply return $\perp$ when it arises. Agreeably, this approach addresses the problem of ambiguity directly, but it is hardly intuitive with respect to the operation of public-key encryption schemes. In particular, this definition is best suited for the class of encryption schemes for which the ambiguity never occurs. However, there is no natural characterisation of this class of schemes.

As a final motivation for a general definition of a strong decryption oracle, let us look at RSA-OAEP [FOPS04]. The non-malleability properties of (a modified version of) this scheme are analysed by Fischlin [Fis05] using a model related to the decryption oracle associated with Equation 1. However, the analysis is restricted to adversaries that only query valid public keys. For such adversaries, the resulting oracle is identical to that resulting from Equation 2, as the decryption algorithm of the scheme checks for key-pair validity and recovers the random coins

used in encryption. However, once this restriction is dropped, the oracles are no longer equivalent. Security with respect to Equation 2 is still implied by Fischlin's analysis but, with respect to Equation 1 it remains an open issue.

SIMPLIFICATION. We now characterise a class of schemes for which the above variants of strong decryption oracle collapse into a simpler definition. This class consists of encryption schemes which perform checks both at encryption and decryption stages. They check for public key validity upon encryption, returning a failure symbol if the key is invalid. Furthermore, in decryption, they check both key-pair validity and that the input ciphertext lies in the range of the encryption algorithm. Note that for such schemes, whenever encryption and decryption do not fail, then correctness ensures that the set of messages which can be obtained using any of the validity criteria above coincide, and have cardinality 1. The simplified version of the strong decryption oracle that we arrive at is shown in Figure 3. The scheme that we present in Section 5 has been designed so that it belongs to this class of encryption schemes, and could therefore be analysed using this simpler oracle. Indeed, this observation is central to our argument that *we propose a simpler and more convenient security model in which to analyse schemes that aim to achieve complete non-malleability*.

procedure $\mathbf{SDecrypt}(\mathsf{c}, \mathsf{PK})$:
$\mathsf{m} \leftarrow_{\$} \{\mathsf{m} : \exists \mathsf{SK}, \mathsf{m} = \mathsf{Dec}(\mathsf{c}, \mathsf{SK}, \mathsf{PK})\}$
Return m

Fig. 3: Simplified definition of strong decryption for schemes which perform all checks. The search over m excludes $\perp$.

4 Security under strong chosen-ciphertext attacks

In this section, we use the general definition of a strong decryption oracle in Figure 1 to extend different security models for encryption schemes. This allows for a uniform treatment of strong security models, some of which have been independently proposed in literature. Then, we investigate the relations among the resulting security notions, as well as those in [Fis05,VV08].

4.1 Indistinguishability of ciphertexts

We now introduce ciphertext indistinguishability under strong chosen-ciphertext attacks as the natural extension of the standard notions of security for public-key encryption schemes. The IND-SCCAx advantage of an adversary $\mathcal{A}$ for $\mathsf{x} = 0, 1, 2$ against a public-key encryption scheme Π is defined by

$$\mathbf{Adv}^{\mathsf{ind\text{-}sccax}}_{\Pi}(\mathcal{A}) := 2 \cdot \Pr\left[\mathsf{IND\text{-}SCCAx}^{\mathcal{A}}_{\Pi} \Rightarrow \mathsf{T}\right] - 1,$$

where game IND-SCCAx is shown in Figure 4. Implicit in this definition are the descriptions of the U and V algorithms, which are fixed when analysing a scheme in the resulting IND-SCCAx model. As seen in the previous section, *one can make general claims of security and still use a simple definition for the strong decryption oracle (Figure 3) by showing that the scheme satisfies a well-defined set of natural properties*.

STRONG PARALLEL ATTACKS. Bellare and Sahai [BS06] define a security notion known as indistinguishability under parallel chosen-ciphertext attacks. Here the adversary can query a vector of ciphertexts to a parallel decryption oracle exactly once and after its left-or-right query, receiving the corresponding component-wise decryptions. It is proved in [BS06] that parallel security maps well to non-malleability of encryption schemes. We extend this model to incorporate strong attacks by defining the IND-SPCAx advantage of an adversary $\mathcal{A}$ against an encryption scheme Π similarly to above, where game IND-SPCAx is shown in Figure 5. Note that under this definition, and consistently with previous results, IND-SPCA2 is equivalent to IND-SCCA2: the parallel oracle is subsumed by the strong decryption oracle that the adversary is allowed to call adaptively after the challenge phase. We remark that

Game $\mathsf{IND\text{-}SCCAx}_{\Pi}$

procedure Initialize():
$\mathsf{I} \leftarrow_{\$} \mathsf{Setup}()$; $(\mathsf{SK}^\star, \mathsf{PK}^\star) \leftarrow_{\$} \mathsf{Gen}()$
$b \leftarrow_{\$} \{0,1\}$; $\mathsf{List} \leftarrow []$; $\mathsf{st}[\mathsf{V}] \leftarrow \mathsf{st}_0$
Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure SDecrypt$(\mathsf{c}, \mathsf{PK}, \mathsf{R})$:
Return $\mathbf{SDecrypt}_{\mathsf{U},\mathsf{V}}(\mathsf{c}, \mathsf{PK}, \mathsf{R})$

procedure Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$:
$\mathsf{c} \leftarrow_{\$} \mathsf{Enc}(\mathsf{m}_b, \mathsf{PK}^\star)$
$\mathsf{List} \leftarrow (\mathsf{c}, \mathsf{PK}^\star) : \mathsf{List}$
Return c

procedure Finalize(b'):
Return $(b' = b)$

Fig. 4: Game defining indistinguishability under strong chosen-ciphertext attacks. An adversary $\mathcal{A}$ is legitimate if: 1) It calls **Left-Right** only once with $\mathsf{m}_0, \mathsf{m}_1 \in \mathsf{MsgSp}(\mathsf{PK})$ such that $|\mathsf{m}_0| = |\mathsf{m}_1|$; and 2) R is polynomial-time and, if $\mathsf{x} = 0$ it does not call **SDecrypt**, if $\mathsf{x} = 1$ it does not call **SDecrypt** after calling **Left-Right**, and if $\mathsf{x} = 2$ it does not call **SDecrypt** with a tuple $(\mathsf{c}, \mathsf{PK})$ in List.

Game $\mathsf{IND\text{-}SPCAx}_{\Pi}$

procedure Initialize():
$\mathsf{I} \leftarrow_{\$} \mathsf{Setup}()$; $(\mathsf{SK}^\star, \mathsf{PK}^\star) \leftarrow_{\$} \mathsf{Gen}()$
$b \leftarrow_{\$} \{0,1\}$; $\mathsf{List} \leftarrow []$; $\mathsf{st}[\mathsf{V}] \leftarrow \mathsf{st}_0$
Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure SDecrypt$(\mathsf{c}, \mathsf{PK}, \mathsf{R})$:
Return $\mathbf{SDecrypt}_{\mathsf{U},\mathsf{V}}(\mathsf{c}, \mathsf{PK}, \mathsf{R})$

procedure Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$:
$\mathsf{c} \leftarrow_{\$} \mathsf{Enc}(\mathsf{m}_b, \mathsf{PK}^\star)$
$\mathsf{List} \leftarrow (\mathsf{c}, \mathsf{PK}^\star) : \mathsf{List}$
Return c

procedure PSDecrypt$(\mathbf{c}, \mathbf{PK}, \mathbf{R})$:
For i from 1 to $\#\mathbf{c}$ do
  $\mathbf{m}[i] \leftarrow_{\$} \mathbf{SDecrypt}_{\mathsf{U},\mathsf{V}}(\mathbf{c}[i], \mathbf{PK}[i], \mathbf{R}[i])$
Return $\mathbf{m}$

procedure Finalize(b'):
Return $(b' = b)$

Fig. 5: Game defining indistinguishability under strong parallel chosen-ciphertext attacks. An adversary $\mathcal{A}$ is legitimate if: 1) It calls **Left-Right** only once with $\mathsf{m}_0, \mathsf{m}_1 \in \mathsf{MsgSp}(\mathsf{PK})$ such that $|\mathsf{m}_0| = |\mathsf{m}_1|$; 2) It calls **PSDecrypt** exactly once and after calling **Left-Right**, on a tuple $(\mathbf{c}, \mathbf{PK}, \mathbf{R})$ such that for $i = 1, \ldots, \#\mathbf{c}$, the tuples $(\mathbf{c}[i], \mathbf{PK}[i])$ do not appear in List and $\mathbf{R}[i]$ are polynomial-time; and 3) R is polynomial-time and, if $\mathsf{x} = 0$ it does not call **SDecrypt**, or if $\mathsf{x} = 1$ it does not call **SDecrypt** after calling **Left-Right**, or if $\mathsf{x} = 2$ it does not call **SDecrypt** with a tuple $(\mathsf{c}, \mathsf{PK})$ in List.

a stronger definition can be adopted, whereby the adversary is allowed to query the parallel oracle with a relation that takes all the ciphertexts simultaneously. We will return to this issue in the next section.

KEM/DEM COMPOSITION. The standard proof technique [CS98] to establish the security of hybrid encryption schemes consisting of a secure keys encapsulation mechanism (KEM) and a secure data encryption mechanism (DEM), fails to extend to the strong chosen-ciphertext models (strong security for KEMs can be defined in the natural way). This failure is due to the non-polynomial nature of the decryption oracle, which cannot be simulated even if one generates the challenge public key. One way to go around this obstacle is to build schemes which permit embedding an *escrow* trapdoor in the common parameters, enabling decryption over *all* public keys.

4.2 Complete non-malleability

Turning our attention to strong notions of non-malleability, or so-called complete non-malleability, we shall see in this section how strong decryption oracles can be used to bring coherence to existing definitional approaches. In particular, we introduce new definitions using strong decryption oracles that can be used to establish clear relations with the strong indistinguishability notion introduced above. We also clarify how the definitions we propose relate to those previously described in literature.

SIMULATION-BASED DEFINITION. The first definition of complete non-malleability was introduced by Fischlin in [Fis05]. It is a simulation-based definition which we translated to the games shown in Figure 9 in Appendix A, for the case where the simulator returns only a public key/ciphertext pair.

We propose an alternative definition. We define the SNM-SCCAx advantage of an adversary $\mathcal{A}$ with respect to a polynomial-time relation R and a polynomial-time simulator $\mathcal{S}$ against a public-key encryption scheme Π by

$$\mathbf{Adv}^{\mathsf{snm\text{-}sccax}}_{\Pi,\mathsf{R},\mathcal{S}}(\mathcal{A}) := \Pr\left[\mathsf{Real\text{-}SNM\text{-}SCCAx}^{\mathcal{A}}_{\Pi,\mathsf{R}} \Rightarrow \mathsf{T}\right] - \Pr\left[\mathsf{Ideal\text{-}SNM\text{-}SCCAx}^{\mathcal{S}}_{\Pi,\mathsf{R}} \Rightarrow \mathsf{T}\right],$$

where games Real-SNM-SCCAx and Ideal-SNM-SCCAx are as shown in Figure 6. The syntax of public-key encryption that we use includes a Setup procedure and hence we explicitly include the common parameters I as

an input to the malleability relation. This approach is consistent with the explicit inclusion of the challenge public key, which is shown in [Fis05] to strictly strengthen the definition. Additionally, for backward compatibility with [BS06], our relations also include the state information st_{R}. For strong decryption oracles that behave consistently with the standard one for $\mathsf{PK}^{\star}$, and for a class of relations that matches those in the original definition, our definition implies standard assisted and non-assisted simulation-based non-malleability as defined in [BS06].

A similar line of reasoning does not permit concluding that our definition also implies Fischlin's complete non-malleability. A legitimate adversary under Fischlin's definition is also a legitimate adversary under the definition in 6. However, we cannot identify a concrete version of the strong decryption oracle that captures the environment under which such an adversary should run. This is because Fischlin's model implicitly uses two definitions of decryption oracle: one during the interactive stages of the game, where the adversary has access to a standard decryption oracle that decrypts using the challenge secret key, and a second one in the **Finalize** stage, where the ciphertext produced by the adversary is decrypted by searching through the message/randomness space. We justify our modelling choice with two arguments. Firstly, the construction of **Finalize** in Fischlin's definition makes it impossible to prove that this security model is stronger than the apparently weaker definition of non-malleability proposed in [BS06], which uses the standard decryption oracle to recover messages from the ciphertexts output by the adversary (recall the particular case of invalid ciphertexts under a valid public key, for which the two interpretations of valid decryption results do not coincide). This suggests that using a consistent definition of a (strong) decryption oracle in all stages of the game is a better approach. Secondly, if this change were introduced in Fischlin's definition, then this would simply be a special case of our more general definition.

Game Real-SNM-SCCAx$_{\Pi,\mathsf{R}}$

procedure Initialize():
$\mathsf{I} \leftarrow_{\$} \mathsf{Setup}()$; $(\mathsf{SK}^{\star}, \mathsf{PK}^{\star}) \leftarrow_{\$} \mathsf{Gen}()$
$\mathsf{List} \leftarrow []$; $\mathsf{st}[\mathsf{V}] \leftarrow \mathsf{st}_0$
Return $(\mathsf{I}, \mathsf{PK}^{\star})$

procedure SDecrypt$(\mathsf{c}, \mathsf{PK}, \mathsf{R}')$:
Return $\mathbf{SDecrypt}_{\mathsf{U},\mathsf{V}}(\mathsf{c}, \mathsf{PK}, \mathsf{R}')$

procedure Encrypt$(\mathrm{M}, \mathsf{st}_{\mathsf{R}})$:
$\mathsf{m} \leftarrow_{\$} \mathrm{M}()$; $\mathsf{c} \leftarrow_{\$} \mathsf{Enc}(\mathsf{m}, \mathsf{PK}^{\star})$
$\mathsf{List} \leftarrow (\mathsf{c}, \mathsf{PK}^{\star}) : \mathsf{List}$
Return c

procedure Finalize$(\mathbf{c}, \mathbf{PK}, \mathbf{R})$:
For i from 1 to $\#\mathbf{c}$ do
  $\mathbf{m}[i] \leftarrow \mathbf{SDecrypt}(\mathbf{c}[i], \mathbf{PK}[i], \mathbf{R}[i])$
Return $\mathsf{R}(\mathsf{I}, \mathsf{m}, \mathbf{m}, \mathbf{c}, \mathsf{PK}^{\star}, \mathbf{PK}, \mathrm{M}, \mathsf{st}_{\mathsf{R}})$

Game Ideal-SNM-SCCAx$_{\Pi,\mathsf{R}}$

procedure Initialize():
$\mathsf{I} \leftarrow_{\$} \mathsf{Setup}()$; $(\mathsf{SK}^{\star}, \mathsf{PK}^{\star}) \leftarrow_{\$} \mathsf{Gen}()$
$\mathsf{st}[\mathsf{V}] \leftarrow \mathsf{st}_0$
Return $(\mathsf{I}, \mathsf{PK}^{\star})$

procedure SDecrypt$(\mathsf{c}, \mathsf{PK}, \mathsf{R}')$:
Return $\mathbf{SDecrypt}_{\mathsf{U},\mathsf{V}}(\mathsf{c}, \mathsf{PK}, \mathsf{R}')$

procedure Finalize$(\mathbf{c}, \mathbf{PK}, \mathbf{R}, \mathrm{M}, \mathsf{st}_{\mathsf{R}})$:
For i from 1 to $\#\mathbf{c}$ do
  $\mathbf{m}[i] \leftarrow \mathbf{SDecrypt}(\mathbf{c}[i], \mathbf{PK}[i], \mathbf{R}[i])$
$\mathsf{m} \leftarrow_{\$} \mathrm{M}()$
Return $\mathsf{R}(\mathsf{I}, \mathsf{m}, \mathbf{m}, \mathbf{c}, \mathsf{PK}^{\star}, \mathbf{PK}, \mathrm{M}, \mathsf{st}_{\mathsf{R}})$

Fig. 6: Games defining simulation-based complete non-malleability under strong chosen-ciphertext attacks. An adversary $\mathcal{A}$, playing the real game, is legitimate if: 1) It calls **Encrypt** once with a valid M; 2) R' queried to **SDecrypt** is computable in polynomial time; if $\mathsf{x} = 0$ it does not call **SDecrypt**; if $\mathsf{x} = 1$ it does not call **SDecrypt** after calling **Left-Right**; and if $\mathsf{x} = 2$ it does not call **SDecrypt** with a tuple in List; and 3) It calls **Finalize** with a tuple such that all relations in $\mathbf{R}$ are computable in polynomial time and, for $i = 1, \ldots, \#\mathbf{c}$, the tuples $(\mathbf{c}[i], \mathbf{PK}[i])$ do not appear in List. A *non-assisted* simulator, playing the ideal game, $\mathcal{S}$ is legitimate if: 1) It calls **Finalize** with a valid M; and 2) It does not call **SDecrypt**. An *assisted* simulator, playing the ideal game, is legitimate if: 1) It calls **Finalize** with a valid M; 2) R' queried to **SDecrypt** is computable in polynomial time; and 3) If $\mathsf{x} = 0$ it does not call **SDecrypt**.

COMPARISON-BASED DEFINITION. The simulation-based definition due to Fischlin was later reformulated by Ventre and Visconti [VV08] as a comparison-based notion. The game underlying this definition is shown in Figure 10 in Appendix A. We introduce an alternative definition based on the CNM-SCCAx game shown in Figure 7 and define CNM-SCCAx advantage of an adversary $\mathcal{A}$ against an encryption scheme Π as

$$\mathbf{Adv}^{\mathsf{cnm\text{-}sccax}}_{\Pi}(\mathcal{A}) := \Pr\left[\mathsf{CNM\text{-}SCCAx}^{\mathcal{A}}_{\Pi} \Rightarrow \mathsf{T} \,\middle|\, b = 1\right] - \Pr\left[\mathsf{CNM\text{-}SCCAx}^{\mathcal{A}}_{\Pi} \Rightarrow \mathsf{T} \,\middle|\, b = 0\right]$$

Our definition differs from that given in [VV08] in the following aspects. We provide the adversary with strong decryption oracles in various stages of the attack. In both models the adversary is allowed to return a vector of ciphertexts, although in [VV08] it is restricted to returning a single public key. Also, procedure $\mathbf{Finalize}$ does not automatically return F if any of the ciphertexts is invalid. The definition in [VV08] would therefore be weaker than ours, were it not for our modelling choice in the $\mathbf{Finalize}$ procedure. In Ventre and Visconti's definition, the relation R is evaluated by a complete search over $(\mathbf{m}[1], r_1) \times \ldots \times (\mathbf{m}[\#\mathbf{c}], r_{\#\mathbf{c}})$. In our definition we have constrained the adversary to performing the search using the strong decryption oracle independently for *each component* in $\mathbf{c}$, before evaluating R. This option is, not only consistent with the standard notions of non-malleability for encryption schemes [BS06], but is also essential to proving equivalence among the different notions we propose.

Game $\mathsf{CNM\text{-}SCCAx}_{\Pi}$

procedure $\mathbf{Initialize}()$:
$\mathsf{I} \leftarrow_{\$} \mathsf{Setup}()$; $(\mathsf{SK}^{\star}, \mathsf{PK}^{\star}) \leftarrow_{\$} \mathsf{Gen}()$
$b \leftarrow_{\$} \{0,1\}$; $\mathsf{List} \leftarrow []$; $\mathsf{st}[\mathsf{V}] \leftarrow \mathsf{st}_0$
Return $(\mathsf{I}, \mathsf{PK}^{\star})$

procedure $\mathbf{SDecrypt}(\mathsf{c}, \mathsf{PK}, \mathsf{R}')$:
Return $\mathbf{SDecrypt}_{\mathsf{U},\mathsf{V}}(\mathsf{c}, \mathsf{PK}, \mathsf{R}')$

procedure $\mathbf{Encrypt}(\mathrm{M})$:
$\mathsf{m}_0, \mathsf{m}_1 \leftarrow_{\$} \mathrm{M}()$
$\mathsf{c} \leftarrow_{\$} \mathsf{Enc}(\mathsf{m}_1, \mathsf{PK}^{\star})$
$\mathsf{List} \leftarrow \mathsf{List} : (\mathsf{c}, \mathsf{PK}^{\star})$
Return c

procedure $\mathbf{Finalize}(\mathbf{c}, \mathbf{PK}, \mathbf{R}, \mathsf{R})$:
For i from 1 to $\#\mathbf{c}$ do
  $\mathbf{m}[i] \leftarrow \mathbf{SDecrypt}(\mathbf{c}[i], \mathbf{PK}[i], \mathbf{R}[i])$
Return $\mathsf{R}(\mathsf{I}, \mathsf{m}_b, \mathbf{m}, \mathbf{c}, \mathsf{PK}^{\star}, \mathbf{PK})$

Fig. 7: Game defining comparison-based complete non-malleability under strong chosen-ciphertext attacks. An adversary $\mathcal{A}$ is legitimate if: 1) It calls $\mathbf{Encrypt}$ once with a valid M; 2) It always queries $\mathbf{SDecrypt}$ with R' computable in polynomial time; if $\mathsf{x} = 0$ it does not call $\mathbf{SDecrypt}$; if $\mathsf{x} = 1$ it does not call $\mathbf{SDecrypt}$ after calling $\mathbf{Left}$-$\mathbf{Right}$; and if $\mathsf{x} = 2$ it does not call $\mathbf{SDecrypt}$ with a tuple $(\mathsf{c}, \mathsf{PK})$ in List; 3) It calls $\mathbf{Finalize}$ with a tuple $(\mathbf{c}, \mathbf{PK}, \mathbf{R}, \mathsf{R})$ such that R and all the elements of $\mathbf{R}$ are computable in polynomial time and, for $i = 1, \ldots, \#\mathbf{c}$, the tuples $(\mathbf{c}[i], \mathbf{PK}[i])$ do not appear in List.

REMARK. Recall that Ventre and Visconti's proof [VV08] of equivalence between comparison and (non-assisted) simulation-based complete non-malleability holds (for $\mathsf{x} \neq 0$) for a restricted class of relations, called *lacking* relations, which do not depend on the challenge public key given to the adversary. We note that our equivalence proof for assisted simulators does not restrict the class of relations under which equivalence holds. Furthermore, such a restriction would be pointless in our definitions for non-assisted simulators, since the proof technique of generating a new key-pair is no longer sufficient to guarantee that the simulator can answer *strong* decryption queries under arbitrary public keys.

4.3 Relations among notions of security

In this section we present a set of theorems that establish equivalence between the security notions we have proposed above. The proofs of these theorems can be found in Appendices B, C and D respectively, and follow the strategy used by Bellare and Sahai in [BS06]. We note that these results hold for *any* instantiation of the general strong decryption oracle we introduced in Figure 1, which provides further evidence that the security models we are relating are, in fact, one and the same notion presented using different formalisms. Using a standard hybrid argument one can show that IND-SPCAx self-composes. Together with our equivalence result, we may conclude that our notions of complete non-malleability also self-compose [PSV07].

Theorem 1 **(IND-SPCAx $\Rightarrow$ CNM-SCCAx).** *Let $\mathcal{A}_{\mathsf{CNM}}$ be a* CNM-SCCAx *adversary against Π. Then there is an* IND-SPCAx *adversary $\mathcal{A}_{\mathsf{IND}}$ against Π with advantage equal to that of $\mathcal{A}_{\mathsf{CNM}}$.*

Theorem 2 **(CNM-SCCAx $\Rightarrow$ SNM-SCCAx).** *Let $\mathcal{A}_{\mathsf{SNM}}$ be a* Real-SNM-SCCAx *adversary against Π and let R be any polynomial-time relation. Then there is an* Ideal-SNM-SCCAx *assisted simulator $\mathcal{S}$ and a* CNM-SCCAx *adversary $\mathcal{A}_{\mathsf{CNM}}$ against Π with advantage equal to that of $\mathcal{A}_{\mathsf{SNM}}$ (with respect to R and $\mathcal{S}$).*

Theorem 3 **(SNM-SCCAx $\Rightarrow$ IND-SPCAx).** *Let $\mathcal{A}_{\mathsf{IND}}$ be an* IND-SPCAx *adversary against Π. Then there is a polynomial-time relation R and a* Real-SNM-SCCAx *adversary $\mathcal{A}_{\mathsf{SNM}}$ such that $\mathcal{A}_{\mathsf{IND}}$'s advantage is upper-bounded by $\mathcal{A}_{\mathsf{SNM}}$'s advantage with respect to any* Ideal-SNM-SCCAx *assisted simulator $\mathcal{S}$.*

5 An efficient completely non-malleable scheme

The only completely non-malleable scheme (without random oracles) known prior to this work, was that of Ventre and Visconti [VV08], which relied on generic (and hence inefficient) zero-knowledge techniques. In this section, we will present an efficient and strongly secure scheme based on standard assumptions.

Our scheme, which is shown in Figure 8, uses a computational bilinear group scheme Γ and a family of hash functions Σ mapping $\mathbb{G}_T \times \mathbb{G} \times \mathbb{G}^2$ to bit strings of size n as described in Appendix E. The scheme's design is based on the certificateless encryption scheme of [DLP08], which in turn is based on Water's identity-based encryption scheme [Wat05]. The construction also uses Water's hash [Wat05], defined by $\mathsf{WH}(w) := u_0 \prod_{i=1}^{n} u_i^{[w]_i}$.

procedure $\mathsf{Setup}_{\Gamma,\Sigma,n}()$:
$\mathsf{k} \leftarrow_\$ \mathsf{Key}(); (\alpha, \beta, u_0, \ldots, u_n) \leftarrow_\$ \mathbb{G}^\star \times \mathbb{G}^{n+2}$
$\mathsf{I} \leftarrow (\Gamma, \mathsf{H}_\mathsf{k}, \alpha, \beta, u_0, \ldots, u_n)$
Return I

procedure $\mathsf{Gen}()$:
$x \leftarrow_\$ \mathbb{Z}_p; X \leftarrow g^x; Y \leftarrow \alpha^x$
$\mathsf{PK} \leftarrow (X, Y); \mathsf{SK} \leftarrow x$
Return $(\mathsf{SK}, \mathsf{PK})$

procedure $\mathsf{Enc}(\mathsf{m}, \mathsf{PK})$:
$t \leftarrow_\$ \mathbb{Z}_p; (X, Y) \leftarrow \mathsf{PK}$
If $\mathbf{e}(X, \alpha) \neq \mathbf{e}(g, Y)$ Return $\perp$
$C_1 \leftarrow \mathsf{m} \cdot \mathbf{e}(Y, \beta^t); C_2 \leftarrow \alpha^t$
$w \leftarrow \mathsf{H}_\mathsf{k}(C_1, C_2, \mathsf{PK})$
$C_3 \leftarrow \mathsf{WH}(w)^t$
$\mathsf{c} \leftarrow (C_1, C_2, C_3)$
Return c

procedure $\mathsf{Dec}(\mathsf{c}, \mathsf{SK}, \mathsf{PK})$:
$(X, Y) \leftarrow \mathsf{PK}$
If $g^{\mathsf{SK}} \neq X \vee \alpha^{\mathsf{SK}} \neq Y$ Return $\perp$
$(C_1, C_2, C_3) \leftarrow \mathsf{c}$
$w \leftarrow \mathsf{H}_\mathsf{k}(C_1, C_2, \mathsf{PK})$
If $\mathbf{e}(C_2, \mathsf{WH}(w)) \neq \mathbf{e}(\alpha, C_3)$ Return $\perp$
$\mathsf{m} \leftarrow C_1 / \mathbf{e}(C_2, \beta^x)$
Return m

Fig. 8: An efficient and strongly secure public-key encryption scheme without random oracles.

VALIDITY ALGORITHMS. We examine which of the validity algorithms exists for this scheme. We assume that Γ specifies algorithms to check for group membership, which are used implicitly throughout the scheme. The MsgSp algorithm is the same as checking membership in $\mathbb{G}_T$. The SKSp algorithm checks membership in $\mathbb{Z}_p$. The KeySp algorithm checks if $g^{\mathsf{SK}} = X$ and $\alpha^{\mathsf{SK}} = Y$ where $(X, Y) = \mathsf{PK}$. The PKSp algorithm checks if $\mathbf{e}(X, \alpha) = \mathbf{e}(g, Y)$. Finally, we show that decryption rejects all ciphertexts outside the range of encryption. Let (C_1, C_2, C_3) be a ciphertext. Then, there exists a message m and a t such that this ciphertext can be written as $(\mathsf{m} \cdot \mathbf{e}(Y, \beta)^t, \alpha^t, C_3)$. If this ciphertext is outside the range of encryption, then $C_3 = \mathsf{WH}(w)^{t'}$ for some $t' \neq t$. But then $\mathbf{e}(C_2, \mathsf{WH}(w)) = \mathbf{e}(\alpha, \mathsf{WH}(w))^t \neq \mathbf{e}(\alpha, \mathsf{WH}(w))^{t'} = \mathbf{e}(\alpha, C_3)$ and the equality check in decryption fails.

The next theorem states the security properties of our scheme. We use the proof technique recently proposed by Bellare and Ristenpart [BR09] and, through a change in the game hopping strategy, we are able to slightly improve the reduction. See Appendix F for full details.

Theorem 4 (Informal). *If the* DBDH *assumption holds and the hash function family* Σ *is collision resistant then the encryption scheme above is* IND-SCCA2 *secure (with respect to definition in Figure 3).*

Although our equivalence theorems imply that this scheme admits a black-box assisted simulator, it does not contradict Fischlin's impossibility results on black-box simulation [Fis05]. First note that Fischlin's impossibility result is in the plain model whereas our scheme has a setup procedure. Furthermore, our definitions do not require the opening of message/randomness pairs, whereas Fischlin requires this to derive his impossibility result for *assisted* simulators. We can indeed construct a non-assisted simulator for our scheme through a direct proof, but this requires modifying the common parameters in an essential way to simulate the strong decryption oracle. Hence this result does not hold for general relations, but only for those which ignore the I presented at their inputs (consistently with [VV08] we call these I-lacking relations). Furthermore, using a similar technique, we are also able to show (through a direct proof) that the zero-knowledge-based construction in [VV08] is completely non-malleable with respect to black-box simulators for a class of relations that are I-lacking (I in this case comprises the common reference string). We note that this is a better result than that obtained in [VV08], since there the class of relations must be both I-lacking and PK-lacking (i.e. they must also ignore the PK at their inputs).

Acknowledgments.

The authors were funded in part by eCrypt II (EU FP7 - ICT-2007-216646) and FCT project PTDC/EIA/71362/2006. The second author was also funded by FCT grant BPD-47924-2008. Research was sponsored by US Army Research laboratory and the UK Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

ABN10. Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 480–497. Springer, 2010.

ARP03. Sattam S. Al-Riyami and Kenneth G. Paterson. Certificateless public key cryptography. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer, 2003.

BBM00. Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000.

BDPR98. Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In Krawczyk [Kra98], pages 26–45.

BF03. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

BK03. Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 491–506. Springer, 2003.

BR06. Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

BR09. Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters' ibe scheme. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 407–424. Springer, 2009.

BS06. Mihir Bellare and Amit Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. Cryptology ePrint Archive, Report 2006/228, 2006. `http://eprint.iacr.org/2006/228`.

Cra08. Ronald Cramer, editor. *Public Key Cryptography - PKC 2008, 11th International Workshop on Practice and Theory in Public-Key Cryptography, Spain, 2008. Proceedings*, volume 4939 of *Lecture Notes in Computer Science*. Springer, 2008.

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Krawczyk [Kra98], pages 13–25.

DDN00. Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM J. Comput.*, 30(2):391–437, 2000.

DLP08. Alexander W. Dent, Benoît Libert, and Kenneth G. Paterson. Certificateless encryption schemes strongly secure in the standard model. In Cramer [Cra08], pages 344–359.

Fis05. Marc Fischlin. Completely non-malleable schemes. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 779–790. Springer, 2005.

FOPS04. Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. Rsa-oaep is secure under the rsa assumption. *J. Cryptology*, 17(2):81–104, 2004.

GM84. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

Kra98. Hugo Krawczyk, editor. *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*. Springer, 1998.

PPV08. Omkant Pandey, Rafael Pass, and Vinod Vaikuntanathan. Adaptive one-way functions and applications. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 57–74. Springer, 2008.

PSV07. Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Relations among notions of non-malleability for encryption. In *ASIACRYPT*, pages 519–535, 2007.

VV08. Carmine Ventre and Ivan Visconti. Completely non-malleable encryption revisited. In Cramer [Cra08], pages 65–84.

Wat05. Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.

A Existing definitions of complete non-malleability

Fis-Real-SNM-SCCAx$_{\Pi,\mathsf{R}}$

procedure Initialize():
$\mathsf{I} \leftarrow_{\$} \mathsf{Setup}(); (\mathsf{SK}^\star, \mathsf{PK}^\star) \leftarrow_{\$} \mathsf{Gen}()$
$\mathsf{List} \leftarrow []$
Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure Decrypt(c):
Return $\mathsf{Dec}(\mathsf{c}, \mathsf{SK}^\star, \mathsf{PK}^\star)$

procedure Encrypt(M):
$\mathsf{m} \leftarrow_{\$} \mathrm{M}(); \mathsf{c} \leftarrow_{\$} \mathsf{Enc}(\mathsf{m}, \mathsf{PK}^\star)$
$\mathsf{List} \leftarrow (\mathsf{c}, \mathsf{PK}^\star) : \mathsf{List}$
Return c

procedure Finalize($\mathsf{c}, \mathsf{PK}, \mathsf{st}_\mathsf{R}$):
If $\exists \mathsf{m}', r\ \mathsf{c} = \mathsf{Enc}(\mathsf{m}', \mathsf{PK}; r) \wedge \mathsf{R}(\mathsf{I}, \mathsf{m}, \mathsf{m}', \mathsf{c}, \mathsf{PK}^\star, \mathsf{PK}, \mathrm{M}, \mathsf{st}_\mathsf{R})$
Return T Else Return F

Fis-Ideal-SNM-SCCAx$_{\Pi,\mathsf{R}}$

procedure Initialize():
$\mathsf{I} \leftarrow_{\$} \mathsf{Setup}(); (\mathsf{SK}^\star, \mathsf{PK}^\star) \leftarrow_{\$} \mathsf{Gen}()$
Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure Decrypt(c):
Return $\mathsf{Dec}(\mathsf{c}, \mathsf{SK}^\star, \mathsf{PK}^\star)$

procedure Finalize($\mathsf{PK}, \mathsf{c}, \mathsf{st}_\mathsf{R}, \mathrm{M}$):
If $\exists \mathsf{m}', r\ \mathsf{c} = \mathsf{Enc}(\mathsf{m}', \mathsf{PK}; r) \wedge \mathsf{R}(\mathsf{I}, \mathsf{m}, \mathsf{m}', \mathsf{c}, \mathsf{PK}^\star, \mathsf{PK}, \mathrm{M}, \mathsf{st}_\mathsf{R})$
Return T Else Return F

Fig. 9: Fischlin's definition of complete non-malleability [Fis05]. The game on top is the real game played by the adversary, and the one at bottom is the ideal game played by a simulator. An adversary $\mathcal{A}$ is legitimate if: 1) It calls **Encrypt** once with a valid M; 2) If $\mathsf{x} = 0$ it does not call **Decrypt**; if $\mathsf{x} = 1$ it does not call **Decrypt** after calling **Left**-**Right**; and if $\mathsf{x} = 2$ it does not call **Decrypt** with a tuple in List; 3) It calls **Finalize** with a tuple $(\mathsf{c}, \mathsf{PK}, \mathsf{st}_\mathsf{R})$ such that tuple $(\mathsf{c}, \mathsf{PK})$ is not in List. A *non-assisted* simulator $\mathcal{S}$ is legitimate if: 1) It calls finalize with a valid M; and 2) It does not call **Decrypt**. An *assisted* simulator is legitimate if: 1) It calls finalize with a valid M; and 2) For $\mathsf{x} = 0$, it does not call **Decrypt** (in this case it is equivalent to a non-assisted simulator).

Fischlin's definition of complete non-malleability adapted to encryption schemes with a Setup procedure is shown in Figure 9. The advantage of an adversary is:

$$\mathbf{Adv}^{\mathsf{fis\text{-}snm\text{-}sccax}}_{\Pi,\mathsf{R},\mathcal{S}}(\mathcal{A}) := \Pr\left[\mathsf{Fis\text{-}Real\text{-}SNM\text{-}SCCAx}^{\mathcal{A}}_{\Pi,\mathsf{R}} \Rightarrow \mathsf{T}\right] - \Pr\left[\mathsf{Fis\text{-}Ideal\text{-}SNM\text{-}SCCAx}^{\mathcal{S}}_{\Pi,\mathsf{R}} \Rightarrow \mathsf{T}\right].$$

VV-CNM-SCCAx$_{\Pi}$

procedure Initialize():
$\mathsf{I} \leftarrow_{\$} \mathsf{Setup}(); (\mathsf{SK}^\star, \mathsf{PK}^\star) \leftarrow_{\$} \mathsf{Gen}()$
$b \leftarrow_{\$} \{0,1\}; \mathsf{List} \leftarrow []$
Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure Decrypt(c):
Return $\mathsf{Dec}(\mathsf{c}, \mathsf{SK}^\star, \mathsf{PK}^\star)$

procedure Encrypt(M):
$\mathsf{m}_0, \mathsf{m}_1 \leftarrow_{\$} \mathrm{M}(); \mathsf{c} \leftarrow_{\$} \mathsf{Enc}(\mathsf{m}_0, \mathsf{PK}^\star)$
$\mathsf{List} \leftarrow (\mathsf{c}, \mathsf{PK}^\star) : \mathsf{List}$
Return c

procedure Finalize($\mathbf{c}, \mathsf{PK}, \mathsf{R}$):
If $\exists \mathbf{m}, r_i$ s.t. $\mathbf{c}[i] = \mathsf{Enc}(\mathbf{m}[i], \mathsf{PK}; r_i) \wedge \mathsf{R}(\mathsf{I}, \mathsf{m}_b, \mathbf{m}, \mathbf{c}, \mathsf{PK}^\star, \mathbf{PK})$
Return T Else Return F

Fig. 10: Definition of complete non-malleability due to Ventre and Visconti [VV08]. An adversary $\mathcal{A}$ is legitimate if: 1) It calls **Encrypt** once with a valid M; 2) If $\mathsf{x} = 0$ it does not call **Decrypt**; if $\mathsf{x} = 1$ it does not call **Decrypt** after calling **Left**-**Right**; and if $\mathsf{x} = 2$ it does not call **Decrypt** with a tuple in List; 3) It calls **Finalize** with a tuple $(\mathbf{c}, \mathsf{PK}, \mathsf{R})$ such that R is computable in polynomial time and for $i = 1, \ldots, \#\mathbf{c}$, the tuples $(\mathbf{c}[i], \mathsf{PK})$ do not appear in List.

The comparison-based definition of complete non-malleability due to Venter and Visconti is shown in Figure 10. The advantage of an adversary is defined by:

$$\mathbf{Adv}^{\mathsf{vv\text{-}cnm\text{-}sccax}}_{\Pi}(\mathcal{A}) := 2 \cdot \Pr\left[\mathsf{VV\text{-}CNM\text{-}SCCAx}^{\mathcal{A}}_{\Pi} \Rightarrow \mathsf{T}\right] - 1.$$

B Proof of Theorem 1: IND-SPCAx $\Rightarrow$ CNM-SCCAx

Proof. Let $\mathcal{A}_{\mathsf{CNM}}$ be a CNM-SCCAx adversary attacking Π. We construct an IND-SPCAx adversary $\mathcal{A}_{\mathsf{IND}}$ attacking Π with the same advantage as shown in Figure 11.

adversary $\mathcal{A}_{\mathsf{IND}}$:
Run $\mathcal{A}_{\mathsf{CNM}}$ in the environment below.

$\mathbf{Initialize}()$:
Get $(\mathsf{I}, \mathsf{PK}^\star)$ from $\mathbf{Initialize}()$
Return $(\mathsf{I}, \mathsf{PK}^\star)$

$\mathbf{Finalize}(\mathbf{c}, \mathbf{PK}, \mathbf{R}, \mathrm{R})$:
Query $\mathbf{PSDecrypt}(\mathbf{c}, \mathbf{PK}, \mathbf{R})$ to get $\mathbf{m}$
If $\mathsf{R}(\mathsf{I}, \mathsf{m}_1, \mathbf{m}, \mathbf{c}, \mathsf{PK}^\star, \mathbf{PK})$ Return 1 Else Return 0

query $\mathbf{SDecrypt}(\mathsf{c}, \mathsf{PK}, \mathsf{R})$:
Query $\mathbf{SDecrypt}(\mathsf{c}, \mathsf{PK}, \mathsf{R})$ to get m
Return m

query $\mathbf{Encrypt}(\mathrm{M})$:
$\mathsf{m}_0, \mathsf{m}_1 \leftarrow_{\$} \mathrm{M}()$
Query $\mathbf{Left\text{-}Right}(\mathsf{m}_0, \mathsf{m}_1)$ to get c
Return c

Fig. 11: An IND-SPCAx adversary based on a CNM-SCCAx adversary.

One can check by examining this adversary that:

$$\Pr\left[\mathsf{IND\text{-}SPCAx}_{\Pi}^{\mathcal{A}_{\mathsf{IND}}} \Rightarrow \mathsf{T} \,\middle|\, b_{\mathsf{IND}} = 1\right] = \Pr\left[\mathsf{CNM\text{-}SCCAx}_{\Pi}^{\mathcal{A}_{\mathsf{CNM}}} \Rightarrow \mathsf{T} \,\middle|\, b_{\mathsf{CNM}} = 1\right], \text{ and}$$
$$\Pr\left[\mathsf{IND\text{-}SPCAx}_{\Pi}^{\mathcal{A}_{\mathsf{IND}}} \Rightarrow \mathsf{T} \,\middle|\, b_{\mathsf{IND}} = 0\right] = \Pr\left[\mathsf{CNM\text{-}SCCAx}_{\Pi}^{\mathcal{A}_{\mathsf{CNM}}} \Rightarrow \mathsf{T} \,\middle|\, b_{\mathsf{CNM}} = 0\right],$$

where b_{IND} and b_{CNM} are the bits chosen in the $\mathbf{Initialize}$ procedures of IND-SPCAx and CNM-SCCAx games respectively, and the theorem follows. □

C Proof of Theorem 2: CNM-SCCAx ⇒ SNM-SCCAx

Proof. Let $\mathcal{A}_{\mathsf{SNM}}$ be a Real-SNM-SCCAx adversary against Π, and $\mathsf{R}_{\mathsf{SNM}}$ a polynomial-time relation. We show that comparison-based non-malleability implies the existence of a canonical *assisted* simulator that satisfies the simulation-based definition of non-malleability. We construct the assisted simulator $\mathcal{S}$ as shown in Figure 12. We then construct a CNM-SCCAx adversary $\mathcal{A}_{\mathsf{CNM}}$ (Figure 13) that, depending on the hidden bit in the CNM-SCCAx game, runs $\mathcal{A}_{\mathsf{SNM}}$ in an environment that is identical to the real game, or to the way it is run by $\mathcal{S}$ in the ideal game. This allows us to show that $\mathcal{A}_{\mathsf{SNM}}$'s advantage with respect to this simulator is the same as $\mathcal{A}_{\mathsf{CNM}}$'s advantage in the CNM-SCCAx game.

adversary $\mathcal{S}$:
Run $\mathcal{A}_{\mathsf{SNM}}$ in the environment below.

$\mathbf{Initialize}()$:
Get $(\mathsf{I}, \mathsf{PK}^\star)$ from $\mathbf{Initialize}()$
Return $(\mathsf{I}, \mathsf{PK}^\star)$

$\mathbf{Finalize}(\mathbf{c}, \mathbf{PK}, \mathbf{R})$:
Return $(\mathbf{c}, \mathbf{PK}, \mathbf{R}, \mathrm{M}, \mathsf{st}_{\mathsf{R}})$

query $\mathbf{SDecrypt}(\mathsf{c}, \mathsf{PK}, \mathsf{R})$:
Query $\mathbf{SDecrypt}(\mathsf{c}, \mathsf{PK}, \mathsf{R})$ to get m
Return m

query $\mathbf{Encrypt}(\mathrm{M}, \mathsf{st}_{\mathsf{R}})$:
$\mathsf{m} \leftarrow_{\$} \mathrm{M}()$
$\mathsf{c} \leftarrow_{\$} \mathsf{Enc}(\mathsf{m}, \mathsf{PK}^\star)$
Return c

Fig. 12: Construction of an Ideal-SNM-SCCAx simulator based on a Real-SNM-CCAx adversary.

Let b denote the bit chosen in the CNM-SCCAx game. One can see from the definition of $\mathcal{A}_{\mathsf{CNM}}$ that:

$$\Pr\left[\mathsf{CNM\text{-}SCCAx}_{\Pi}^{\mathcal{A}_{\mathsf{CNM}}} \Rightarrow \mathsf{T} \,\middle|\, b = 1\right] = \Pr\left[\mathsf{Real\text{-}SNM\text{-}SCCAx}_{\Pi,\mathsf{R}_{\mathsf{SNM}}}^{\mathcal{A}_{\mathsf{SNM}}} \Rightarrow \mathsf{T}\right].$$

This is because if $b = 1$, in both cases the relation will be evaluated on the message which is encapsulated under the challenge ciphertext given to $\mathcal{A}_{\mathsf{SNM}}$.

By substituting the simulator $\mathcal{S}$ in the $\mathsf{Ideal\text{-}SNM\text{-}SCCAx}_{\Pi,\mathsf{R}_{\mathsf{SNM}}}$ game we also see that:

$$\Pr\left[\mathsf{CNM\text{-}SCCAx}_{\Pi}^{\mathcal{A}_{\mathsf{CNM}}} \Rightarrow \mathsf{T} \,\middle|\, b = 0\right] = \Pr\left[\mathsf{Ideal\text{-}SNM\text{-}SCCAx}_{\Pi,\mathsf{R}_{\mathsf{SNM}}}^{\mathcal{S}} \Rightarrow \mathsf{T}\right].$$

adversary $\mathcal{A}_{\mathsf{CNM}}$:
Run $\mathcal{A}_{\mathsf{SNM}}$ in the environment below.

$\mathbf{Initialize}()$:
Get $(\mathsf{I}, \mathsf{PK}^\star)$ from $\mathbf{Initialize}()$
Return $(\mathsf{I}, \mathsf{PK}^\star)$

$\mathbf{Finalize}(\mathbf{c}, \mathbf{PK}, \mathbf{R})$:
$\mathsf{R}_{\mathsf{CNM}}(\mathsf{I}, \mathsf{m}, \mathbf{m}, \mathbf{c}, \mathsf{PK}, \mathbf{PK}) := \mathsf{R}_{\mathsf{SNM}}(\mathsf{I}, \mathsf{m}, \mathbf{m}, \mathbf{c}, \mathsf{PK}, \mathbf{PK}, \mathrm{M}, \mathsf{st}_{\mathsf{R}})$
Return $(\mathbf{c}, \mathbf{PK}, \mathbf{R}, \mathsf{R}_{\mathsf{CNM}})$

query $\mathbf{SDecrypt}(\mathsf{c}, \mathsf{PK}, \mathsf{R})$:
Query $\mathbf{SDecrypt}(\mathsf{c}, \mathsf{PK}, \mathsf{R})$ to get m
Return m

query $\mathbf{Encrypt}(\mathrm{M}, \mathsf{st}_{\mathsf{R}})$:
Query $\mathbf{Encrypt}(\mathrm{M})$ to get c
Return c

Fig. 13: Construction of a CNM-SCCAx adversary based on a Real-SNM-SCCAx adversary.

This is because if $b = 0$, in both cases the relation will be evaluated on a message which is chosen independently of the challenge ciphertext given to $\mathcal{A}_{\mathsf{SNM}}$.

The results follows by subtracting the above equalities. □

D Proof of Theorem 3: SNM-SCCAx ⇒ IND-SPCAx

Proof. We prove the theorem for $\mathsf{x} = 2$. The theorem can be proved using techniques in [BS06] for the $\mathsf{x} = 0, 1$. Let $\mathcal{A}_{\mathsf{IND}}$ be an IND-SPCA2 adversary attacking Π. We construct a Real-SNM-SCCA2 adversary $\mathcal{A}_{\mathsf{SNM}}$ attacking Π as shown in Figure 14. This adversary is geared to work with the relation R as shown in Figure 15. Note that appending 0 to the re-encrypted messages ensures that $\{\mathsf{m}_0, \mathsf{m}_1\} \cap \{0\mathsf{m}_0, 0\mathsf{m}_1\}$ is empty and hence the returned ciphertext is new.

adversary $\mathcal{A}_{\mathsf{SNM}}$:
Run $\mathcal{A}_{\mathsf{IND}}$ in the environment below.

$\mathbf{Initialize}()$:
Get $(\mathsf{I}, \mathsf{PK}^\star)$ from $\mathbf{Initialize}()$
Return $(\mathsf{I}, \mathsf{PK}^\star)$

$\mathbf{Finalize}(b')$:
$\mathbf{c}[1] \leftarrow_{\$} \mathsf{Enc}(0\mathsf{m}_{b'}, \mathsf{PK}^\star)$
$\mathbf{PK}[1] \leftarrow \mathsf{PK}^\star$
$\mathbf{R}[1] := \mathsf{T}$
Return $(\mathbf{c}, \mathbf{PK}, \mathbf{R})$

query $\mathbf{SDecrypt}(\mathsf{c}, \mathsf{PK}, \mathsf{R})$:
Query $\mathbf{SDecrypt}(\mathsf{c}, \mathsf{PK}, \mathsf{R})$ to get m
Return m

query $\mathbf{PSDecrypt}(\mathbf{c}, \mathbf{PK}, \mathbf{R})$:
For i from 1 to $\#\mathbf{c}$
  Query $\mathbf{SDecrypt}(\mathbf{c}[i], \mathbf{PK}[i], \mathbf{R}[i])$ to get $\mathbf{m}[i]$
Return $\mathbf{m}$

query $\mathbf{Left\text{-}Right}(\mathsf{m}_0, \mathsf{m}_1)$:
$\mathrm{M} \leftarrow \{\mathsf{m}_0, \mathsf{m}_1\}$; $\mathsf{st}_{\mathsf{R}} \leftarrow (\mathsf{m}_0, \mathsf{m}_1)$
Query $\mathbf{Encrypt}(\mathrm{M}, \mathsf{st}_{\mathsf{R}})$ to get c
Return c

Fig. 14: Construction of an IND-SPCAx adversary based on a Real-SNM-SCCAx adversary.

relation $\mathsf{R}(\mathsf{I}, \mathsf{m}, \mathbf{m}, \mathbf{c}, \mathsf{PK}, \mathbf{PK}, \mathrm{M}, \mathsf{st}_{\mathsf{R}})$:
$(\mathsf{m}_0, \mathsf{m}_1) \leftarrow \mathsf{st}_{\mathsf{R}}$
If $\mathsf{m}_0 = \mathsf{m}_1 \vee \mathrm{M} \neq \{\mathsf{m}_0, \mathsf{m}_1\}$ Return F
If $\#\mathbf{PK} \neq 1 \vee \mathbf{PK}[1] \neq \mathsf{PK} \vee \#\mathbf{m} \neq 1$ Return F
If $\mathbf{m}[1] = 0\mathsf{m}$ Return T Else Return F

Fig. 15: The relation R.

We claim that for the relation in Figure 15 we have:

$$\Pr\left[\mathsf{Real\text{-}SNM\text{-}SCCAx}^{\mathcal{A}_{\mathsf{SNM}}}_{\Pi,\mathsf{R}} \Rightarrow \mathsf{T}\right] = \Pr\left[\mathsf{IND\text{-}SPCAx}^{\mathcal{A}_{\mathsf{IND}}}_{\Pi} \Rightarrow \mathsf{T}\right].$$

This can be seen from Figures 14 and 15 and noting that in game IND-SPCAx we may restrict the adversary to output two *distinct* messages with no loss in advantage. This tells us that the value computed by the relation identifies the challenge message.

We also claim that for any simulator $\mathcal{S}$ and for the relation in Figure 15 we have that:

$$\Pr\left[\mathsf{Ideal\text{-}SNM\text{-}CCA}x^{\mathcal{S}}_{\Pi,\mathsf{R}} \Rightarrow 1\right] \leq \frac{1}{2}$$

This holds since the simulator gets no information about the encrypted plaintexts, and the probability that it satisfies the other conditions of relation R is at most 1.

The result now follows from the above two equations and using definitions of advantage. □

E Bilinear groups and collision resistance hashing

BILINEAR GROUPS. We say $\Gamma = (\mathbb{G}, \mathbb{G}_T, p, \mathbf{e})$ is a (symmetric) bilinear group scheme if: 1) $\mathbb{G}$, and $\mathbb{G}_T$ are groups of prime order p; and 2) $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a non-degenerate, efficiently computable bilinear map [BF03]. For a group $\mathbb{G}$, we define $\mathbb{G}^\star := \mathbb{G} \setminus \{\mathbf{1}\}$ where $\mathbf{1}$ is the identity element of $\mathbb{G}$ with respect to its group operation.

THE DECISIONAL BILINEAR DIFFIE-HELLMAN PROBLEM. The DBDH advantage of an adversary $\mathcal{A}$ against Γ is

$$\mathbf{Adv}^{\mathsf{dbdh}}_{\Gamma}(\mathcal{A}) := 2 \cdot \Pr\left[\mathsf{DBDH}^{\mathcal{A}}_{\Gamma} \Rightarrow \mathsf{T}\right] - 1.$$

where game DBDH is shown in Figure 16.

Game DBDH_Γ

procedure Initialize():
$a, b, c \leftarrow_\$ \mathbb{Z}_p$; $d \leftarrow_\$ \{0,1\}$
If $d = 1$ Then $W \leftarrow \mathbf{e}(g,g)^{abc}$ Else $W \leftarrow_\$ \mathbb{G}_T$
Return (g^a, g^b, g^c, W)

procedure Finalize(d'):
Return $(d' = d)$

Fig. 16: Game defining the decisional bilinear Diffie-Hellman problem.

COLLISION RESISTANT HASH FAMILY. A hash family $\Sigma = (\mathsf{Key}, \mathsf{H})$ is defined via a pair of probabilistic polynomial-time algorithms as follows. Key is the probabilistic key generation algorithm which outputs a key k. Algorithm H takes as input a key k and a string $w \in \{0,1\}^\star$ and outputs a string $\mathsf{H}_\mathsf{k}(w) \in \{0,1\}^\ell$ for a polynomial $\ell(\cdot)$.

The CR advantage of an adversary $\mathcal{A}$ against hash family Σ is defined by

$$\mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}) := \Pr\left[\mathsf{CR}^{\mathcal{A}}_{\Sigma} \Rightarrow \mathsf{T}\right].$$

where CR is the game shown in Figure 17.

Game CR_Σ

procedure Initialize():
$\mathsf{k} \leftarrow_\$ \mathsf{Key}()$
Return k

procedure Finalize(w, w'):
If $(w' \neq w) \wedge \mathsf{H}_\mathsf{k}(w) = \mathsf{H}_\mathsf{k}(w')$ Return T Else Return F

Fig. 17: Game defining collision resistance.

F Proof of Theorem 4: DBDH $\wedge$ CR $\Rightarrow$ IND-SCCA2

First, we give a precise statement of the theorem.

Theorem 5. *Let Γ be a bilinear group description, $n \geq 1$ an integer, Σ a family of hash functions, and Π the encryption scheme in Figure 8 instantiated for these parameters. Let $\mathcal{A}$ be an* IND-SCCA2 *adversary with advantage $\epsilon > 0$. Suppose $\mathcal{A}$ makes at most $Q_{\mathsf{Dec}} \in [1 \ldots p\epsilon/6n)$ strong decryption queries. Then there exist a* DBDH *adversary $\mathcal{A}'$ against Γ and a* CR *adversary $\mathcal{A}''$ against Σ such that:*

$$\mathbf{Adv}^{\mathsf{dbdh}}_{\Gamma}(\mathcal{A}') + 1/2\mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}'') \geq \frac{\epsilon^2}{18nQ_{\mathsf{Dec}} + 3\epsilon}.$$

Proof. The proof proceeds by a sequence of games Game_0 to Game_5. All games involve an attacker $\mathcal{A}$ who attempts to guess a hidden bit d for which she eventually outputs a guess d'. For all i, we call Win_i the event that $\mathcal{A}$ is successful (i.e. that $d' = d$) in Game_i. Formally, this matches the case in which the corresponding game returns 1. We also call BD^j_i to the event that Game_i sets the Bad_j flag.

Before presenting the game sequence, we introduce the following definitions. Let $m = \lceil 6Q_{\mathsf{Dec}}/\epsilon \rceil$, where Q_{Dec} and ϵ are as stated in the Theorem formulation. Also, let $\mathbf{x} = (x_0, \ldots, x_n) \in \mathcal{X}$, where $\mathcal{X} = [-n(m-1)..0] \times ([0..m-1])^n$ and $\mathbf{y} = (y_0, \ldots, y_n) \in \mathcal{Y}$ where $\mathcal{Y} = \mathbb{Z}_p^{n+1}$. Finally, for $w \in \{0,1\}^n$, let

$$\mathsf{F}(\mathbf{x}, w) = x_0 + \sum_{i=1}^{n} x_i[w]_i \qquad\qquad \mathsf{G}(\mathbf{y}, w) = y_0 + \sum_{i=1}^{n} y_i[w]_i \mod p$$

Note that while the computation of G above is over $\mathbb{Z}_p$, that of F is over $\mathbb{Z}$.

Game_0*:* In this game, shown on the left side of Figure 18, $\mathcal{A}$ is interacting with the IND-SCCA2 attack environment (the Bad flags cause no effect in the execution). Note that, following the security model definition, we assume that the environment can answer decryption queries without knowing the matching secret values for changed public keys. We also assume without loss of generality that the adversary performs exactly Q_{Dec} decryption queries, all of them different. Let $D = \{(\mathsf{c}_1, \mathsf{PK}_1, w_1), \ldots, (\mathsf{c}_{Q_{\mathsf{Dec}}}, \mathsf{PK}_{Q_{\mathsf{Dec}}}, w_{Q_{\mathsf{Dec}}})\}$ be the list of tuples involved in these decryption queries. For this game, we have

$$2 \cdot \Pr[\mathsf{Win}_0] - 1 = \epsilon$$

Game_1*:* This game is identical to Game_0 except that:

1. The environment generates α and β differently by making $\alpha = g^\gamma$ and $\beta = g^b$, for random $\gamma \leftarrow \mathbb{Z}_p^\star$ and $b \in \mathbb{Z}_p$. Note that, since the distribution of α and β is identical to that in the previous game, the way these parameters are generated does not affect the analysis.
2. Two of the cases where environment sets the Bad_1 flag now cause extra code to be executed: the environment will select a random d' and use this to calculate its final output. These cases correspond to the adversary causing a collision in the H_k function.

Lemma 1. *There exists a PPT adversary $\mathcal{A}''$ such that:*

$$\Pr[\mathsf{BD}^1_0] = \Pr[\mathsf{BD}^1_1] = \mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}'')$$

Furthermore, we have:

$$\Pr[\mathsf{Win}_0 \wedge \neg\mathsf{BD}^1_0] = \Pr[\mathsf{Win}_1 \wedge \neg\mathsf{BD}^1_1]$$

Game_2*:* Here, we change the code after the setting of Bad_2. For this hop, we must follow the same line of reasoning as described in [BR09], which permits relating the adversary's probability of success in both games by finding upper and lower bounds for $\Pr[\neg\mathsf{BD}_1^2|\neg\mathsf{BD}_1^1]$, which are very close to each-other.

Lemma 2. *There exists a PPT adversary $\mathcal{A}''$ such that:*

$$2\cdot\Pr[\mathsf{Win}_2]-1\geq\frac{2}{n(m-1)+1}(1-\frac{Q_{\mathsf{Dec}}}{m})\Pr[\mathsf{Win}_0]-\frac{1}{n(m-1)+1}-\mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}'')$$

Game_3*:* In this game we rearrange the generation of the common parameters and move the setting of Bad_2 to the points in the game where the relevant hash values are calculated. These changes have no influence in the distribution of the game's output or in the probability of occurrence of BD events. Note that we now have $\mathsf{H}(w)=\beta^{\mathsf{F}(\mathbf{x},w)}g^{\mathsf{G}(\mathbf{y},w)}$.

Game_4*:* We now change the way in which the challenge is generated, and also the also the algorithm for calculating answers to decryption queries. We note that this means that this game is identical to the previous one, unless BD_4^2 occurs. This means that $\Pr[\mathsf{Win}_4\wedge\neg\mathsf{BD}_4^2]=\Pr[\mathsf{Win}_3\wedge\neg\mathsf{BD}_3^2]=\Pr[\mathsf{Win}_2\wedge\neg\mathsf{BD}_2^2]$, and also $\Pr[\mathsf{BD}_4^2]=\Pr[\mathsf{BD}_3^2]=\Pr[\mathsf{BD}_2^2]$. Because of the structure of the games, which forces $\Pr[\mathsf{Win}_i|\mathsf{BD}_i^j]=1/2$ this means that $\Pr[\mathsf{Win}_4]=\Pr[\mathsf{Win}_3]=\Pr[\mathsf{Win}_2]$.

Game_5*:* We finally change the way the environment generates the challenge public key and the challenge ciphertext. The entire game environment now depends on random elements $A,B,C\in\mathbb{G}$ and also on a random element $T\in\mathbb{G}_T$. Call a,b,c to the (unknown) discrete logarithms of A,B,C with respect to g. Observe that, if $T=e(g^a,g^b)^c$ then Game_5 is in fact identical to Game_4. Furthermore, since the challenge ciphertext now holds no information on the encrypted message, as T operates as a one-time pad, we must have $\Pr[\mathsf{Win}_5]=1/2$.

Lemma 3. *There exists a PPT adversary $\mathcal{A}'$ such that:*

$$\Pr[\mathsf{Win}_4]-1/2=\mathbf{Adv}^{\mathsf{dbdh}}_{\Gamma}(\mathcal{A}').$$

We now combine the various inequalities, using $\delta=\frac{1}{n(m-1)+1}$.

$$\Pr[\mathsf{Win}_2]-1/2=\mathbf{Adv}^{\mathsf{dbdh}}_{\Gamma}(\mathcal{A}')$$

$$\mathbf{Adv}^{\mathsf{dbdh}}_{\Gamma}(\mathcal{A}')+1/2\mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}'')\geq\delta(1-\frac{Q_{\mathsf{Dec}}}{m})\Pr[\mathsf{Win}_0]-\frac{\delta}{2}$$

$$\mathbf{Adv}^{\mathsf{dbdh}}_{\Gamma}(\mathcal{A}')+1/2\mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}'')\geq\delta(1-\frac{Q_{\mathsf{Dec}}}{m})(\frac{\epsilon}{2}+\frac{1}{2})-\frac{\delta}{2}$$

$$\mathbf{Adv}^{\mathsf{dbdh}}_{\Gamma}(\mathcal{A}')+1/2\mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}'')\geq\frac{\delta}{2}[(1-\frac{Q_{\mathsf{Dec}}}{m})\epsilon-\frac{Q_{\mathsf{Dec}}}{m}]$$

We now take advantage of the fact that $\epsilon\leq 1$ and $m=\lceil 6Q_{\mathsf{Dec}}/\epsilon\rceil\geq 6Q_{\mathsf{Dec}}/\epsilon$ to derive a simpler expression for the reduction:

$$\mathbf{Adv}^{\mathsf{dbdh}}_{\Gamma}(\mathcal{A}')+1/2\mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}'')\geq\frac{\delta\epsilon}{12}(6-\epsilon-1)=\frac{\delta\epsilon}{3}$$

Finally, substituting for δ and using $m=\lceil 6Q_{\mathsf{Dec}}/\epsilon\rceil\leq 6Q_{\mathsf{Dec}}/\epsilon+1$, we have

$$\mathbf{Adv}^{\mathsf{dbdh}}_{\Gamma}(\mathcal{A}')+1/2\mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}'')\geq\frac{\epsilon}{3}\frac{1}{n(m-1)+1}\geq\frac{\epsilon^2}{18nQ_{\mathsf{Dec}}+3\epsilon}$$

□

Proof (Lemma 1). First note that the two games are identical unless BD_1^1 occurs in lines 141 or 143. The second part of the lemma and also $\Pr[\mathsf{BD}_1^1] = \Pr[\mathsf{BD}_0^1]$ follows directly from this observation and the results in [BR09]. To obtain the first part of the lemma, we construct an algorithm $\mathcal{A}''$ that uses the adversary $\mathcal{A}$ to produce a collision for $\mathsf{H_k}$ as follows:

- It runs the adversary according to the rules of Game_1, changing only the way in which line `027` is executed. Instead of searching for the (m, r) pair, the algorithm takes advantage of its knowledge of the discrete logarithm b. Since it is only required to decrypt ciphertexts under well-formed public keys (i.e. a public key $\mathsf{PK} = (X, Y)$ where $\mathbf{e}(X, \alpha) = \mathbf{e}(g, Y)$), we can assume that $X = g^x$ and $Y = \alpha^x$ for some (possibly unknown) value x. Hence, the decryption oracle will work correctly by calculating:

$$\mathsf{m} = \frac{C_1}{\mathbf{e}(C_2, X^b)} = \frac{C_1}{\mathbf{e}(C_2, \beta^x)}$$

  Note that this means that algorithm $\mathcal{A}''$ executes in polynomial time, as required.
- When Bad_1 is set in either line 141 or line 143, the algorithm returns the two $\mathsf{H_k}$ inputs that originated the collision.

We present the description of the algorithm in Figure 20, from which it is possible to infer the execution time of the adversary indicated in the theorem statement.

From the algorithm description we have that $\Pr[\mathsf{BD}_1^1] = \mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}'')$, and the lemma follows.

□

Proof (Lemma 2). We start by noting that the probability of BD_2^2 is dependent on the particular set of decryption queries placed by the adversary, represented by list D, and also on the challenge ciphertext. To deal with this, we start by observing that the two games are identical until BD_2^2 happens, which means that independently of the particular sequence of adversarial queries, we have:

$$\Pr[\mathsf{Win}_2 \wedge \neg\mathsf{BD}_2^2] = \Pr[\mathsf{Win}_1 \wedge \neg\mathsf{BD}_1^2]$$

$$\Pr[\mathsf{BD}_1^2] = \Pr[\mathsf{BD}_2^2]$$

We can also write:

$$\begin{aligned}
\Pr[\mathsf{Win}_2] &= \Pr[\mathsf{Win}_2 \wedge \neg\mathsf{BD}_2^2] + \Pr[\mathsf{Win}_2 \wedge \mathsf{BD}_2^2] \\
&= \Pr[\mathsf{Win}_1 \wedge \neg\mathsf{BD}_1^2] + 1/2 \cdot \Pr[\mathsf{BD}_1^2] \\
&= \Pr[\mathsf{Win}_1 \wedge \neg\mathsf{BD}_1^2 \wedge \mathsf{BD}_1^1] + \Pr[\mathsf{Win}_1 \wedge \neg\mathsf{BD}_1^2 \wedge \neg\mathsf{BD}_1^1] + 1/2 \cdot \Pr[\mathsf{BD}_1^2 \wedge \mathsf{BD}_1^1] + 1/2 \cdot \Pr[\mathsf{BD}_1^2 \wedge \neg\mathsf{BD}_1^1] \\
&= \Pr[\mathsf{Win}_1 \wedge \neg\mathsf{BD}_1^2 \wedge \neg\mathsf{BD}_1^1] + \Pr[\mathsf{BD}_1^1] + 1/2 \cdot \Pr[\mathsf{BD}_1^2 \wedge \neg\mathsf{BD}_1^1] \\
&= \Pr[\mathsf{Win}_1 \wedge \neg\mathsf{BD}_1^2 | \neg\mathsf{BD}_1^1] \Pr[\neg\mathsf{BD}_1^1] + 1/2 \cdot \Pr[\mathsf{BD}_1^2 | \neg\mathsf{BD}_1^1] \Pr[\neg\mathsf{BD}_1^1] + \Pr[\mathsf{BD}_1^1] \\
&= (1 - \mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}''))(\Pr[\mathsf{Win}_1 \wedge \neg\mathsf{BD}_1^2 | \neg\mathsf{BD}_1^1] + 1/2 - 1/2 \cdot \Pr[\neg\mathsf{BD}_1^2 | \neg\mathsf{BD}_1^1]) + \mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}'')
\end{aligned}$$

Hence:

$$2 \cdot \Pr[\mathsf{Win}_2] - 1 \geq 2 \cdot \Pr[\mathsf{Win}_1 \wedge \neg\mathsf{BD}_1^2 | \neg\mathsf{BD}_1^1] - \Pr[\neg\mathsf{BD}_1^2 | \neg\mathsf{BD}_1^1] - \mathbf{Adv}^{\mathsf{cr}}_{\Sigma}(\mathcal{A}'')$$

Defining event W as the occurrence of a particular list of hash values $(w^\star, w_0, \ldots, w_1)$ due to adversarial queries, conditional independence [BR09] allows us to write

$$\Pr[\mathsf{Win}_1 \wedge \neg\mathsf{BD}_1^2 \wedge \mathsf{W} | \neg\mathsf{BD}_1^1] = \Pr[\neg\mathsf{BD}_1^2 | \neg\mathsf{BD}_1^1] \Pr[\mathsf{Win}_1 \wedge \mathsf{W} | \neg\mathsf{BD}_1^1]$$

$$\Pr[\neg\mathsf{BD}_1^2 \wedge \mathsf{W} | \neg\mathsf{BD}_1^1] = \Pr[\neg\mathsf{BD}_1^2 | \neg\mathsf{BD}_1^1] \Pr[\mathsf{W} | \neg\mathsf{BD}_1^1]$$

In turn, summing these probabilities over all possible $(w^\star, w_0, \ldots, w_1)$, this allows us to re-write our equations above for all possible adversarial queries as

$$2 \cdot \Pr[\mathsf{Win}_2] - 1 \geq 2 \cdot \Pr_{\min}[\neg \mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1] \Pr[\mathsf{Win}_1 | \neg \mathsf{BD}_1^1] - \Pr_{\max}[\neg \mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1] - \mathbf{Adv}_{\Sigma}^{\mathsf{cr}}(\mathcal{A}'')$$

Now, using the results from Lemma 1, we can establish that

$$2 \cdot \Pr[\mathsf{Win}_2] - 1 \geq 2 \cdot \Pr_{\min}[\neg \mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1] \Pr[\mathsf{Win}_0 | \neg \mathsf{BD}_0^1] - \Pr_{\max}[\neg \mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1] - \mathbf{Adv}_{\Sigma}^{\mathsf{cr}}(\mathcal{A}'')$$

$$2 \cdot \Pr[\mathsf{Win}_2] - 1 \geq 2 \cdot \Pr_{\min}[\neg \mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1] \Pr[\mathsf{Win}_0] - \Pr_{\max}[\neg \mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1] - \mathbf{Adv}_{\Sigma}^{\mathsf{cr}}(\mathcal{A}'')$$

since in Game_0 the occurrence of BD_0^1 does not affect the adversary's probability of success in any way.

To finally obtain the Lemma, we need to provide appropriate lower and upper bounds for $\Pr[\neg \mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1]$. We start with the lower bound, by writing

$$\Pr[\neg \mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1] = \Pr[\mathsf{F}(\mathbf{x}, w^\star) = 0 \bigwedge_{(\cdot,\cdot,w) \in D} \mathsf{F}(\mathbf{x}, w) \neq 0 | \neg \mathsf{BD}_1^1]$$

We can rewrite this as

$$\Pr[\mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1] = \Pr[\mathsf{F}(\mathbf{x}, w^\star) \neq 0 \bigvee_{(\cdot,\cdot,w) \in D} \mathsf{F}(\mathbf{x}, w) = 0 | \neg \mathsf{BD}_1^1]$$

$$\Pr[\mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1] \leq \Pr[\mathsf{F}(\mathbf{x}, w^\star) \neq 0 | \neg \mathsf{BD}_1^1] + \sum_{(\cdot,\cdot,w) \in D} \Pr[\mathsf{F}(\mathbf{x}, w^\star) = 0 \wedge \mathsf{F}(\mathbf{x}, w) = 0 | \neg \mathsf{BD}_1^1]$$

Following the same reasoning as in [BR09], which takes advantage of the fact that, since BD_1^1 does not occur, each w must differ from $w^\star$ in at least one bit, we have that

$$\Pr[\neg \mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1] \geq \frac{1}{n(m-1)+1}(1 - \frac{=}{m}) = \Pr_{\min}[\neg \mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1]$$

$$\Pr[\neg \mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1] \leq \frac{1}{n(m-1)+1} = \Pr_{\max}[\neg \mathsf{BD}_1^2 | \neg \mathsf{BD}_1^1]$$

And the Lemma follows.

□

Proof (Lemma 3). We construct an algorithm $\mathcal{A}'$ that interpolates between Game_4 and Game_5 and solves the DBDH problem with an advantage which is identical to $\mathcal{A}$'s capability in distinguishing the two games. $\mathcal{A}'$ works exactly like Game_5, except that: (1) it takes (A, B, C, T) to be the challenge parameters in the DBDH game; and (2) it simulates the strong decryption oracle by also applying lines 426* the decryption procedure in line 027 (note that this means that our algorithm works in polynomial time, as required). We present the algorithm in Figure 20, from which it is possible to infer the execution time of the adversary indicated in the theorem statement.

It is clear that if $T = e(A, B)^c$ (call this event DBDH), where $C = g^c$, then our algorithm is running the adversary under the rules of Game_4. Conversely, if T is a random element in $\mathbb{G}_T$, then the algorithm is simulating the environment of Game_5. This means that we can write

$$\Pr[d = d' \rightarrow 1 | \mathtt{DBDH}] = \Pr[\mathsf{Win}_4]$$

$$\Pr[d = d' | \neg \mathtt{DBDH}] = \Pr[\mathsf{Win}_5]$$

and the Lemma follows from

$$\mathbf{Adv}_{\Gamma}^{\mathsf{dbdh}}(\mathcal{A}') = \Pr[d = d' | \mathtt{DBDH}] + \Pr[d \neq d' | \neg \mathtt{DBDH}] - 1$$

$$\mathbf{Adv}_{\Gamma}^{\mathsf{dbdh}}(\mathcal{A}') = \Pr[\mathsf{Win}_4] - \Pr[\mathsf{Win}_5]$$

□

procedure Initialize(): Game_0

000 $\mathsf{k} \leftarrow \mathsf{Key}()$
001 $\alpha \leftarrow \mathbb{G}^\star$
002 $\beta \leftarrow \mathbb{G}$
003 For $j = 0, \ldots, n$ do
004 $\quad u_i \leftarrow \mathbb{G}$
005 $\mathsf{I} \leftarrow (\Gamma, \mathsf{H}_k, \alpha, \beta, u_0, \ldots, u_n)$
006 $x^\star \leftarrow \mathbb{Z}_p$; $X^\star \leftarrow g^{x^\star}$; $Y^\star \leftarrow \alpha^{x^\star}$
007 $\mathsf{PK}^\star \leftarrow (X^\star, Y^\star)$; $\mathsf{SK}^\star \leftarrow (x^\star, \mathsf{PK}^\star)$
008 $D \leftarrow \{\}$
009 Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure SDecrypt$(\mathsf{c}, \mathsf{PK})$: Game_0

020 $w \leftarrow \mathsf{H}_\mathsf{k}(C_1, C_2, \mathsf{PK})$
021 Add $(\mathsf{c}, \mathsf{PK}, w)$ to D
022 Parse $(X, Y) \leftarrow \mathsf{PK}$
023 If $\mathbf{e}(X, \alpha) \neq \mathbf{e}(g, Y)$ Return $\perp$
024 Parse $(C_1, C_2, C_3) \leftarrow \mathsf{c}$
025 If $\mathbf{e}(C_2, \mathsf{H}(w)) \neq \mathbf{e}(\alpha, C_3)$ Return $\perp$
026 If $\mathsf{PK} = \mathsf{PK}^\star$ set $\mathsf{m} \leftarrow C_1 / \mathbf{e}(C_2, \beta^{x^\star})$
027 Else find (r, m) s.t. $\mathsf{Enc}(\mathsf{m}, \mathsf{PK}; r) = \mathsf{c}$
028 Return m

procedure Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$: Game_0

030 $d \leftarrow \{0,1\}$; $t^\star \leftarrow \mathbb{Z}_p$
031 $C_1^\star \leftarrow \mathsf{m}_d \cdot \mathbf{e}(Y^\star, \beta^{t^\star})$; $C_2^\star \leftarrow \alpha^{t^\star}$
032 $w^\star \leftarrow \mathsf{H}_\mathsf{k}(C_1, C_2, \mathsf{PK}^\star)$
033 $C_3^\star \leftarrow \mathsf{H}(w^\star)^{t^\star}$
034 $\mathsf{c}^\star \leftarrow (C_1^\star, C_2^\star, C_3^\star)$
035 Return $\mathsf{c}^\star$

procedure Finalize(d'): Game_0

040 If $(C, \mathsf{PK}, w^\star) \in D \wedge (\mathsf{c}, \mathsf{PK}) \neq (\mathsf{c}^\star, \mathsf{PK}^\star)$
041 $\quad$ Set Bad_1
042 If $\exists (\cdot, \cdot, w_i), (\cdot, \cdot, w_j) \in D \mid i \neq j \wedge w_i = w_j$
043 $\quad$ Set Bad_1
044 $\mathbf{x} \leftarrow \mathcal{X}$
045 If $\mathsf{F}(w^\star, \mathbf{x}) \neq 0 \bigvee_{i=1}^{Q_{\mathsf{Dec}}} \mathsf{F}(w_i, \mathbf{x}) = 0$
046 $\quad$ Set Bad_2
047 Return $(d' = d)$

procedure Initialize(): Game_1

000 $\mathsf{k} \leftarrow \mathsf{Key}()$
101 $\gamma \leftarrow \mathbb{Z}_p^\star$; $\alpha \leftarrow g^\gamma$
102 $b \leftarrow \mathbb{Z}_p$; $\beta \leftarrow g^b$
003 For $j = 0, \ldots, n$ do
004 $\quad u_i \leftarrow \mathbb{G}$
005 $\mathsf{I} \leftarrow (\Gamma, \mathsf{H}_\mathsf{k}, \alpha, \beta, u_0, \ldots, u_n)$
006 $x^\star \leftarrow \mathbb{Z}_p$; $X^\star \leftarrow g^{x^\star}$; $Y^\star \leftarrow \alpha^{x^\star}$
007 $\mathsf{PK}^\star \leftarrow (X^\star, Y^\star)$; $\mathsf{SK}^\star \leftarrow (x^\star, \mathsf{PK}^\star)$
008 $D \leftarrow \{\}$
009 Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure SDecrypt$(\mathsf{c}, \mathsf{PK})$: Game_1

020 $w \leftarrow \mathsf{H}_\mathsf{k}(C_1, C_2, \mathsf{PK})$
021 Add $(\mathsf{c}, \mathsf{PK}, w)$ to D
022 Parse $(X, Y) \leftarrow \mathsf{PK}$
023 If $\mathbf{e}(X, \alpha) \neq \mathbf{e}(g, Y)$ Return $\perp$
024 Parse $(C_1, C_2, C_3) \leftarrow \mathsf{c}$
025 If $\mathbf{e}(C_2, \mathsf{H}(w)) \neq \mathbf{e}(\alpha, C_3)$ Return $\perp$
026 If $\mathsf{PK} = \mathsf{PK}^\star$ set $\mathsf{m} \leftarrow C_1 / \mathbf{e}(C_2, \beta^{x^\star})$
027 Else find (r, m) s.t. $\mathsf{Enc}(\mathsf{m}, \mathsf{PK}; r) = \mathsf{c}$
028 Return m

procedure Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$: Game_1

030 $d \leftarrow \{0,1\}$; $t^\star \leftarrow \mathbb{Z}_p$
031 $C_1^\star \leftarrow \mathsf{m}_d \cdot \mathbf{e}(Y^\star, \beta^{t^\star})$; $C_2^\star \leftarrow \alpha^{t^\star}$
032 $w^\star \leftarrow \mathsf{H}_\mathsf{k}(C_1, C_2, \mathsf{PK}^\star)$
033 $C_3^\star \leftarrow \mathsf{H}(w^\star)^{t^\star}$
034 $\mathsf{c}^\star \leftarrow (C_1^\star, C_2^\star, C_3^\star)$
035 Return $\mathsf{c}^\star$

procedure Finalize(d'): Game_1

040 If $(C, \mathsf{PK}, w^\star) \in D \wedge (\mathsf{c}, \mathsf{PK}) \neq (\mathsf{c}^\star, \mathsf{PK}^\star)$
141 $\quad$ Set Bad_1; $d' \leftarrow \{0,1\}$
042 If $\exists (\cdot, \cdot, w_i), (\cdot, \cdot, w_j) \in D \mid i \neq j \wedge w_i = w_j$
143 $\quad$ Set Bad_1; $d' \leftarrow \{0,1\}$
044 $\mathbf{x} \leftarrow \mathcal{X}$
045 If $\mathsf{F}(w^\star, \mathbf{x}) \neq 0 \bigvee_{i=1}^{Q_{\mathsf{Dec}}} \mathsf{F}(w_i, \mathbf{x}) = 0$
046 $\quad$ Set Bad_2
047 Return $(d' = d)$

procedure Initialize(): Game_2

000 $\mathsf{k} \leftarrow \mathsf{Key}()$
101 $\gamma \leftarrow \mathbb{Z}_p^\star$; $\alpha \leftarrow g^\gamma$
102 $b \leftarrow \mathbb{Z}_p$; $\beta \leftarrow g^b$
003 For $j = 0, \ldots, n$ do
004 $\quad u_i \leftarrow \mathbb{G}$
005 $\mathsf{I} \leftarrow (\Gamma, \mathsf{H}_\mathsf{k}, \alpha, \beta, u_0, \ldots, u_n)$
006 $x^\star \leftarrow \mathbb{Z}_p$; $X^\star \leftarrow g^{x^\star}$; $Y^\star \leftarrow \alpha^{x^\star}$
007 $\mathsf{PK}^\star \leftarrow (X^\star, Y^\star)$; $\mathsf{SK}^\star \leftarrow (x^\star, \mathsf{PK}^\star)$
008 $D \leftarrow \{\}$
009 Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure SDecrypt$(\mathsf{c}, \mathsf{PK})$: Game_2

020 $w \leftarrow \mathsf{H}_\mathsf{k}(C_1, C_2, \mathsf{PK})$
021 Add $(\mathsf{c}, \mathsf{PK}, w)$ to D
022 Parse $(X, Y) \leftarrow \mathsf{PK}$
023 If $\mathbf{e}(X, \alpha) \neq \mathbf{e}(g, Y)$ Return $\perp$
024 Parse $(C_1, C_2, C_3) \leftarrow \mathsf{c}$
025 If $\mathbf{e}(C_2, \mathsf{H}(w)) \neq \mathbf{e}(\alpha, C_3)$ Return $\perp$
026 If $\mathsf{PK} = \mathsf{PK}^\star$ set $\mathsf{m} \leftarrow C_1 / \mathbf{e}(C_2, \beta^{x^\star})$
027 Else find (r, m) s.t. $\mathsf{Enc}(\mathsf{m}, \mathsf{PK}; r) = \mathsf{c}$
028 Return m

procedure Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$: Game_2

030 $d \leftarrow \{0,1\}$; $t^\star \leftarrow \mathbb{Z}_p$
031 $C_1^\star \leftarrow \mathsf{m}_d \cdot \mathbf{e}(Y^\star, \beta^{t^\star})$; $C_2^\star \leftarrow \alpha^{t^\star}$
032 $w^\star \leftarrow \mathsf{H}_\mathsf{k}(C_1, C_2, \mathsf{PK}^\star)$
033 $C_3^\star \leftarrow \mathsf{H}(w^\star)^{t^\star}$
034 $\mathsf{c}^\star \leftarrow (C_1^\star, C_2^\star, C_3^\star)$
035 Return $\mathsf{c}^\star$

procedure Finalize(d'): Game_2

040 If $(C, \mathsf{PK}, w^\star) \in D \wedge (\mathsf{c}, \mathsf{PK}) \neq (\mathsf{c}^\star, \mathsf{PK}^\star)$
141 $\quad$ Set Bad_1; $d' \leftarrow \{0,1\}$
042 If $\exists (\cdot, \cdot, w_i), (\cdot, \cdot, w_j) \in D \mid i \neq j \wedge w_i = w_j$
143 $\quad$ Set Bad_1; $d' \leftarrow \{0,1\}$
044 $\mathbf{x} \leftarrow \mathcal{X}$
045 If $\mathsf{F}(w^\star, \mathbf{x}) \neq 0 \bigvee_{i=1}^{Q_{\mathsf{Dec}}} \mathsf{F}(w_i, \mathbf{x}) = 0$
246 $\quad$ Set Bad_2; $d' \leftarrow \{0,1\}$
047 Return $(d' = d)$

Fig. 18: Games Game_0, Game_1, and Game_2.

procedure Initialize(): Game_3

000 $\mathsf{k} \leftarrow \mathsf{Key}()$
301 $\gamma \leftarrow \mathbb{Z}_p^\star$; $\alpha \leftarrow g^\gamma$; $\mathbf{x} \leftarrow \mathcal{X}$
302 $b \leftarrow \mathbb{Z}_p$; $\beta \leftarrow g^b$; $\mathbf{y} \leftarrow \mathcal{Y}$
003 For $j = 0, \ldots, n$ do
304 $\quad u_j \leftarrow \beta^{x_j} g^{y_j}$
005 $\mathsf{I} \leftarrow (\Gamma, \mathsf{H_k}, \alpha, \beta, u_0, \ldots, u_n)$
006 $x^\star \leftarrow \mathbb{Z}_p$; $X^\star \leftarrow g^{x^\star}$; $Y^\star \leftarrow \alpha^{x^\star}$
007 $\mathsf{PK}^\star \leftarrow (X^\star, Y^\star)$; $\mathsf{SK}^\star \leftarrow (x^\star, \mathsf{PK}^\star)$
008 $D \leftarrow \{\}$
009 Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure SDecrypt$(\mathsf{c}, \mathsf{PK})$: Game_3

320 $w \leftarrow \mathsf{H_k}(C_1, C_2, \mathsf{PK})$; If $\mathsf{F}(w, \mathbf{x}) = 0$ Set Bad_2
021 Add $(\mathsf{c}, \mathsf{PK}, w)$ to D
022 Parse $(X, Y) \leftarrow \mathsf{PK}$
023 If $\mathbf{e}(X, \alpha) \neq \mathbf{e}(g, Y)$ Return $\perp$
024 Parse $(C_1, C_2, C_3) \leftarrow \mathsf{c}$
025 If $\mathbf{e}(C_2, \mathsf{H}(w)) \neq \mathbf{e}(\alpha, C_3)$ Return $\perp$
026 If $\mathsf{PK} = \mathsf{PK}^\star$ set $\mathsf{m} \leftarrow C_1 / \mathbf{e}(C_2, \beta^{x^\star})$
027 Else find (r, m) s.t. $\mathsf{Enc}(\mathsf{m}, \mathsf{PK}; r) = \mathsf{c}$
028 Return m

procedure Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$: Game_3

030 $d \leftarrow \{0, 1\}$; $t^\star \leftarrow \mathbb{Z}_p$
031 $C_1^\star \leftarrow \mathsf{m}_d \cdot \mathbf{e}(Y^\star, \beta^{t^\star})$; $C_2^\star \leftarrow \alpha^{t^\star}$
332 $w^\star \leftarrow \mathsf{H_k}(C_1, C_2, \mathsf{PK}^\star)$; If $\mathsf{F}(w^\star, \mathbf{x}) \neq 0$ Set Bad_2
033 $C_3^\star \leftarrow \mathsf{H}(w^\star)^{t^\star}$
034 $\mathsf{c}^\star \leftarrow (C_1^\star, C_2^\star, C_3^\star)$
035 Return $\mathsf{c}^\star$

procedure Finalize(d'): Game_3

040 If $(C, \mathsf{PK}, w^\star) \in D \wedge (\mathsf{c}, \mathsf{PK}) \neq (\mathsf{c}^\star, \mathsf{PK}^\star)$
141 $\quad$ Set Bad_1; $d' \leftarrow \{0, 1\}$
042 If $\exists (\cdot, \cdot, w_i), (\cdot, \cdot, w_j) \in D \mid i \neq j \wedge w_i = w_j$
143 $\quad$ Set Bad_1; $d' \leftarrow \{0, 1\}$
344
345
346 If Bad_2 then $d' \leftarrow \{0, 1\}$
047 Return $(d' = d)$

procedure Initialize(): Game_4

000 $\mathsf{k} \leftarrow \mathsf{Key}()$
301 $\gamma \leftarrow \mathbb{Z}_p^\star$; $\alpha \leftarrow g^\gamma$; $\mathbf{x} \leftarrow \mathcal{X}$
302 $b \leftarrow \mathbb{Z}_p$; $\beta \leftarrow g^b$; $\mathbf{y} \leftarrow \mathcal{Y}$
003 For $j = 0, \ldots, n$ do
304 $\quad u_j \leftarrow \beta^{x_j} g^{y_j}$
005 $\mathsf{I} \leftarrow (\Gamma, \mathsf{H_k}, \alpha, \beta, u_0, \ldots, u_n)$
006 $x^\star \leftarrow \mathbb{Z}_p$; $X^\star \leftarrow g^{x^\star}$; $Y^\star \leftarrow \alpha^{x^\star}$
007 $\mathsf{PK}^\star \leftarrow (X^\star, Y^\star)$; $\mathsf{SK}^\star \leftarrow (x^\star, \mathsf{PK}^\star)$
008 $D \leftarrow \{\}$
009 Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure SDecrypt$(\mathsf{c}, \mathsf{PK})$: Game_4

320 $w \leftarrow \mathsf{H_k}(C_1, C_2, \mathsf{PK})$; If $\mathsf{F}(w, \mathbf{x}) = 0$ Set Bad_2
021 Add $(\mathsf{c}, \mathsf{PK}, w)$ to D
022 Parse $(X, Y) \leftarrow \mathsf{PK}$
023 If $\mathbf{e}(X, \alpha) \neq \mathbf{e}(g, Y)$ Return $\perp$
024 Parse $(C_1, C_2, C_3) \leftarrow \mathsf{c}$
025 If $\mathbf{e}(C_2, \mathsf{H}(w)) \neq \mathbf{e}(\alpha, C_3)$ Return $\perp$
4261 If $\mathsf{PK} = \mathsf{PK}^\star$
4262 $\quad$ If Bad_2 set $\mathsf{m} = \perp$
4263 $\quad$ Else $Z = C_3 / C_2^{\mathsf{G}(\mathbf{y}, w)/\gamma}$
4264 $\quad\quad \mathsf{m} \leftarrow C_2 / \mathbf{e}(Y, Z^{1/\mathsf{F}(\mathbf{x}, w)})$
027 Else find (r, m) s.t. $\mathsf{Enc}(\mathsf{m}, \mathsf{PK}; r) = \mathsf{c}$
028 Return m

procedure Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$: Game_4

030 $d \leftarrow \{0, 1\}$; $t^\star \leftarrow \mathbb{Z}_p$
031 $C_1^\star \leftarrow \mathsf{m}_d \cdot \mathbf{e}(Y^\star, \beta^{t^\star})$; $C_2^\star \leftarrow \alpha^{t^\star}$
332 $w^\star \leftarrow \mathsf{H_k}(C_1, C_2, \mathsf{PK}^\star)$; If $\mathsf{F}(w^\star, \mathbf{x}) \neq 0$ Set Bad_2
4331 $\quad$ If Bad_2 set $C_3^\star = \perp$
4332 $\quad$ Else $C_3^\star \leftarrow (g^{\mathsf{G}(\mathbf{y}, w^\star)})^{t^\star}$
034 $\mathsf{c}^\star \leftarrow (C_1^\star, C_2^\star, C_3^\star)$
035 Return $\mathsf{c}^\star$

procedure Finalize(d'): Game_4

040 If $(C, \mathsf{PK}, w^\star) \in D \wedge (\mathsf{c}, \mathsf{PK}) \neq (\mathsf{c}^\star, \mathsf{PK}^\star)$
141 $\quad$ Set Bad_1; $d' \leftarrow \{0, 1\}$
042 If $\exists (\cdot, \cdot, w_i), (\cdot, \cdot, w_j) \in D \mid i \neq j \wedge w_i = w_j$
143 $\quad$ Set Bad_1; $d' \leftarrow \{0, 1\}$
344
345
346 If Bad_2 then $d' \leftarrow \{0, 1\}$
047 Return $(d' = d)$

procedure Initialize(): Game_5

000 $\mathsf{k} \leftarrow \mathsf{Key}()$
301 $\gamma \leftarrow \mathbb{Z}_p^\star$; $\alpha \leftarrow g^\gamma$; $\mathbf{x} \leftarrow \mathcal{X}$
502 $B \leftarrow \mathbb{G}$; $\beta \leftarrow B$; $\mathbf{y} \leftarrow \mathcal{Y}$
003 For $j = 0, \ldots, n$ do
304 $\quad u_j \leftarrow \beta^{x_j} g^{y_j}$
005 $\mathsf{I} \leftarrow (\Gamma, \mathsf{H_k}, \alpha, \beta, u_0, \ldots, u_n)$
506 $A \leftarrow \mathbb{G}$; $X^\star \leftarrow A$; $Y^\star \leftarrow A^\gamma$
507 $\mathsf{PK}^\star \leftarrow (X^\star, Y^\star)$; $\mathsf{SK}^\star \leftarrow (\perp, \mathsf{PK}^\star)$
008 $D \leftarrow \{\}$
009 Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure SDecrypt$(\mathsf{c}, \mathsf{PK})$: Game_5

320 $w \leftarrow \mathsf{H_k}(C_1, C_2, \mathsf{PK})$; If $\mathsf{F}(w, \mathbf{x}) = 0$ Set Bad_2
021 Add $(\mathsf{c}, \mathsf{PK}, w)$ to D
022 Parse $(X, Y) \leftarrow \mathsf{PK}$
023 If $\mathbf{e}(X, \alpha) \neq \mathbf{e}(g, Y)$ Return $\perp$
024 Parse $(C_1, C_2, C_3) \leftarrow \mathsf{c}$
025 If $\mathbf{e}(C_2, \mathsf{H}(w)) \neq \mathbf{e}(\alpha, C_3)$ Return $\perp$
4261 If $\mathsf{PK} = \mathsf{PK}^\star$
4262 $\quad$ If Bad_2 set $\mathsf{m} = \perp$
4263 $\quad$ Else $Z = C_3 / C_2^{\mathsf{G}(\mathbf{y}, w)/\gamma}$
4264 $\quad\quad \mathsf{m} \leftarrow C_2 / \mathbf{e}(Y, Z^{1/\mathsf{F}(\mathbf{x}, w)})$
027 Else find (r, m) s.t. $\mathsf{Enc}(\mathsf{m}, \mathsf{PK}; r) = \mathsf{c}$
028 Return m

procedure Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$: Game_5

530 $d \leftarrow \{0, 1\}$; $C \leftarrow \mathbb{G}$; $T \leftarrow \mathbb{G}_T$
531 $C_1^\star \leftarrow \mathsf{m}_d \cdot T$; $C_2^\star \leftarrow C^\gamma$
332 $w^\star \leftarrow \mathsf{H_k}(C_1, C_2, \mathsf{PK}^\star)$; If $\mathsf{F}(w^\star, \mathbf{x}) \neq 0$ Set Bad_2
4331 $\quad$ If Bad_2 set $C_3^\star = \perp$
5332 $\quad$ Else $C_3^\star \leftarrow C^{\mathsf{G}(\mathbf{y}, w^\star)}$
034 $\mathsf{c}^\star \leftarrow (C_1^\star, C_2^\star, C_3^\star)$
035 Return $\mathsf{c}^\star$

procedure Finalize(d'): Game_5

040 If $(C, \mathsf{PK}, w^\star) \in D \wedge (\mathsf{c}, \mathsf{PK}) \neq (\mathsf{c}^\star, \mathsf{PK}^\star)$
141 $\quad$ Set Bad_1; $d' \leftarrow \{0, 1\}$
042 If $\exists (\cdot, \cdot, w_i), (\cdot, \cdot, w_j) \in D \mid i \neq j \wedge w_i = w_j$
143 $\quad$ Set Bad_1; $d' \leftarrow \{0, 1\}$
344
345
346 If Bad_2 then $d' \leftarrow \{0, 1\}$
047 Return $(d' = d)$

Fig. 19: Games Game_3, Game_4, and Game_5.

procedure Initialize():

100 $\mathsf{k} \leftarrow \mathbf{Initialize}()$
101 $\gamma \leftarrow \mathbb{Z}_p^\star$; $\alpha \leftarrow g^\gamma$
102 $b \leftarrow \mathbb{Z}_p$; $\beta \leftarrow g^b$
003 For $j = 0, \ldots, n$ do
004 $\quad u_i \leftarrow \mathbb{G}$
005 $\mathsf{I} \leftarrow (\Gamma, \mathsf{H}_\mathsf{k}, \alpha, \beta, u_0, \ldots, u_n)$
006 $x^\star \leftarrow \mathbb{Z}_p$; $X^\star \leftarrow g^{x^\star}$; $Y^\star \leftarrow \alpha^{x^\star}$
007 $\mathsf{PK}^\star \leftarrow (X^\star, Y^\star)$; $\mathsf{SK}^\star \leftarrow (x^\star, \mathsf{PK}^\star)$
008 $D \leftarrow \{\}$
009 Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure SDecrypt$(\mathsf{c}, \mathsf{PK})$:

020 $w \leftarrow \mathsf{H}_\mathsf{k}(C_1, C_2, \mathsf{PK})$
021 Add $(\mathsf{c}, \mathsf{PK}, w)$ to D
022 Parse $(X, Y) \leftarrow \mathsf{PK}$
023 If $\mathbf{e}(X, \alpha) \neq \mathbf{e}(g, Y)$ Return $\perp$
024 Parse $(C_1, C_2, C_3) \leftarrow \mathsf{c}$
025 If $\mathbf{e}(C_2, \mathsf{H}(w)) \neq \mathbf{e}(\alpha, C_3)$ Return $\perp$
026 If $\mathsf{PK} = \mathsf{PK}^\star$ set $\mathsf{m} \leftarrow C_1 / \mathbf{e}(C_2, \beta^{x^\star})$
027 Else $\mathsf{m} \leftarrow C_1 / \mathbf{e}(C_2, X^b)$
028 Return m

procedure Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$:

030 $d \leftarrow \{0,1\}$; $t^\star \leftarrow \mathbb{Z}_p$
031 $C_1^\star \leftarrow \mathsf{m}_d \cdot \mathbf{e}(Y^\star, \beta^{t^\star})$; $C_2^\star \leftarrow \alpha^{t^\star}$
032 $w^\star \leftarrow \mathsf{H}_\mathsf{k}(C_1, C_2, \mathsf{PK}^\star)$
033 $C_3^\star \leftarrow \mathsf{H}(w^\star)^{t^\star}$
034 $\mathsf{c}^\star \leftarrow (C_1^\star, C_2^\star, C_3^\star)$
035 Return $\mathsf{c}^\star$

procedure Finalize(d'):

040 If $(C, \mathsf{PK}, w^\star) \in D \wedge (\mathsf{c}, \mathsf{PK}) \neq (\mathsf{c}^\star, \mathsf{PK}^\star)$
141 $\quad$ Set Bad_1; $\mathbf{Finalize}((C, \mathsf{PK}), (C^\star, \mathsf{PK}^\star))$
042 If $\exists (C_i, \mathsf{PK}_i, w_i), (C_j, \mathsf{PK}_j, w_j) \in D \mid i \neq j \wedge w_i = w_j$
143 $\quad$ Set Bad_1; $\mathbf{Finalize}((C_i, \mathsf{PK}_i), (C_j, \mathsf{PK}_j))$
044 $\mathbf{x} \leftarrow \mathcal{X}$
045 If $\mathsf{F}(w^\star, \mathbf{x}) \neq 0 \bigvee_{i=1}^{Q_{\mathrm{Dec}}} \mathsf{F}(w_i, \mathbf{x}) = 0$
046 $\quad$ Set Bad_2
047 $\mathbf{Finalize}(\perp, \perp)$

procedure Initialize():

000 $(A, B, C, T) \leftarrow \mathbf{Initialize}()$; $\mathsf{k} \leftarrow \mathsf{Key}()$
301 $\gamma \leftarrow \mathbb{Z}_p^\star$; $\alpha \leftarrow g^\gamma$; $\mathbf{x} \leftarrow \mathcal{X}$
502 $\beta \leftarrow B$; $\mathbf{y} \leftarrow \mathcal{Y}$
003 For $j = 0, \ldots, n$ do
304 $\quad u_j \leftarrow \beta^{x_j} g^{y_j}$
005 $\mathsf{I} \leftarrow (\Gamma, \mathsf{H}_\mathsf{k}, \alpha, \beta, u_0, \ldots, u_n)$
506 $X^\star \leftarrow A$; $Y^\star \leftarrow A^\gamma$
507 $\mathsf{PK}^\star \leftarrow (X^\star, Y^\star)$; $\mathsf{SK}^\star \leftarrow (\perp, \mathsf{PK}^\star)$
008 $D \leftarrow \{\}$
009 Return $(\mathsf{I}, \mathsf{PK}^\star)$

procedure SDecrypt$(\mathsf{c}, \mathsf{PK})$:

320 $w \leftarrow \mathsf{H}_\mathsf{k}(C_1, C_2, \mathsf{PK})$; If $\mathsf{F}(w, \mathbf{x}) = 0$ Set Bad_2
021 Add $(\mathsf{c}, \mathsf{PK}, w)$ to D
022 Parse $(X, Y) \leftarrow \mathsf{PK}$
023 If $\mathbf{e}(X, \alpha) \neq \mathbf{e}(g, Y)$ Return $\perp$
024 Parse $(C_1, C_2, C_3) \leftarrow \mathsf{c}$
025 If $\mathbf{e}(C_2, \mathsf{H}(w)) \neq \mathbf{e}(\alpha, C_3)$ Return $\perp$
4262 If Bad_2 set $\mathsf{m} = \perp$
4263 Else $Z = C_3 / C_2^{\mathsf{G}(\mathbf{y}, w)/\gamma}$
4264 $\quad \mathsf{m} \leftarrow C_2 / \mathbf{e}(Y, Z^{1/\mathsf{F}(\mathbf{x}, w)})$
028 Return m

procedure Left-Right$(\mathsf{m}_0, \mathsf{m}_1)$:

530 $d \leftarrow \{0,1\}$
531 $C_1^\star \leftarrow \mathsf{m}_d \cdot T$; $C_2^\star \leftarrow C^\gamma$
332 $w^\star \leftarrow \mathsf{H}_\mathsf{k}(C_1, C_2, \mathsf{PK}^\star)$; If $\mathsf{F}(w^\star, \mathbf{x}) \neq 0$ Set Bad_2
4331 $\quad$ If Bad_2 set $C_3^\star = \perp$
5332 $\quad$ Else $C_3^\star \leftarrow C^{\mathsf{G}(\mathbf{y}, w^\star)}$
034 $\mathsf{c}^\star \leftarrow (C_1^\star, C_2^\star, C_3^\star)$
035 Return $\mathsf{c}^\star$

procedure Finalize(d'):

040 If $(C, \mathsf{PK}, w^\star) \in D \wedge (\mathsf{c}, \mathsf{PK}) \neq (\mathsf{c}^\star, \mathsf{PK}^\star)$
141 $\quad$ Set Bad_1; $d' \leftarrow \{0,1\}$
042 If $\exists (\cdot, \cdot, w_i), (\cdot, \cdot, w_j) \in D \mid i \neq j \wedge w_i = w_j$
143 $\quad$ Set Bad_1; $d' \leftarrow \{0,1\}$
346 If Bad_2 then $d' \leftarrow \{0,1\}$
047 $\mathbf{Finalize}(d' = d)$

Fig. 20: The CR adversary $\mathcal{A}''$ (left) and the DBDH adversary $\mathcal{A}'$ (right).