Selective Opening Security from Simulatable Data Encapsulation*

Felix Heuer and Bertram Poettering

Horst Görtz Institute for IT Security
Ruhr University Bochum, Bochum, Germany
{felix.heuer,bertram.poettering}@rub.de

Abstract. The confidentiality notion of security against selective opening attacks considers adversaries that obtain challenge ciphertexts and are allowed to adaptively open them, thereby revealing the encrypted message and the randomness used to encrypt. The SO notion is stronger than that of CCA security and is often required when formally arguing towards the security of multi-user applications. While different ways of achieving correspondingly secure schemes are known, as they generally employ expensive asymmetric building blocks like lossy trapdoor functions or lossy encryption, such constructions are routinely left aside by practitioners and standardization bodies. So far, formal arguments towards the SO security of schemes used in practice (e.g., for email encryption) are not known.
In this work we shift the focus from the asymmetric to the symmetric building blocks of PKE and prove the following statement: If a PKE scheme is composed of a key encapsulation mechanism (KEM) and a blockcipher-based data encapsulation mechanism (DEM), and the DEM meets specific combinatorial properties, then the PKE scheme offers SO security, in the ideal cipher model. Fortunately, as we show, the required properties hold for popular modes of operation like CTR, CBC, CCM, and GCM. This paper not only establishes the corresponding theoretical framework of analysis, but also contributes very concretely to practical cryptography by concluding that selective opening security is given for many real-world schemes.

1 Introduction

Public key encryption in the multi-user setting. The most important security notion for public key encryption is indistinguishability under chosen ciphertext attacks (IND-CCA). The modeled setting is as follows: One user generates a key pair, a second users encrypts one out of two messages to her, and the adversary shall find out which one it was. Here, importantly, the adversary controls the distribution of the two messages and may request decryptions of ciphertexts of its choice.

The definition of selective opening (SO) security is more general as it takes into account the fact that the public key setting allows for more than two parties. Concretely, in the SO setting one user generates a key pair, many users encrypt messages to her key (of course using fresh and independent random coins), and the adversary's goal is to derive any information about any of the messages. Again the adversary controls the message distribution (individually for each participant, but also joint distributions are possible) and may have arbitrary ciphertexts decrypted. On top of that the adversary is allowed to 'open' any subset of ciphertexts, i.e., to corrupt the encrypters, for instance by breaking into their computers, and thereby reveal the messages they encrypted and the random coins they used. (In some applications, like in secure multi-party computation, users even deliberately reveal their messages and randomness to make their computations publicly verifiable.) Selective opening security is provided if in this situation the confidentiality of the remaining 'unopened' ciphertexts is still provided. Intuitively, as all the encryptions occur independently of each other, IND-CCA should imply SO security. Unfortunately, formal analysis reveals that this is not the case.

Notions of Selective Opening security. Formalising suitable notions of SO security has proven to be highly non-trivial. Since encrypted messages may depend on each other, opening some ciphertexts might readily leak information on messages encrypted in other (unopened) ciphertexts. Thus, it is not even clear what it means for unopened messages to remain confidential. Two flavours of SO security have been studied in prior work: notions based on indistinguishability (IND) and notions based on simulatability (SIM).

* An extended abstract of this paper appears in the proceedings of ASIACRYPT 2016. This is the full version.

For IND based notions an adversary may open arbitrary ciphertexts and is challenged to tell apart the originally encrypted messages from fresh messages that occur *as likely as the original messages*. One usually restricts the distribution on the messages to be *efficiently conditionally resamplable* to ensure an efficient security game (*weak*-IND-SO). We obtain the security experiment for *full*-IND-SO if arbitrary distributions may occur in the experiment.

In contrast, SIM based notions do not suffer from such a restriction. In a nutshell, a scheme is SIM-SO secure if for every SO adversary there exists a simulator that can compute the same output without seeing any ciphertexts. Importantly, such simulators may corrupt senders to learn the messages they (virtually) encrypted.

Both flavours may be considered for passive (CPA) and active (CCA) adversaries whereby, in contrast to the CPA setting, a CCA adversary has access to a decryption oracle (with the usual restrictions). While any of IND-SO-CPA/CCA and SIM-SO-CPA/CCA implies standard IND-CPA/CCA security, the converse does not hold in general. Only partial results are known for the reverse direction, as discussed below. We give more details on the relations amongst the notions of selective opening security at the end of Section 2.

Motivation and contribution. Considering that users in practice may be exposed to the threats modeled in the SO context, and given that the classical indistinguishability notions are formally weaker than notions of SO security, the following question is immediate: Are users 'safe' if they trust in a PKE scheme designed towards the goal of 'only' indistinguishability? At least in theory, if the security proof of the scheme considers exclusively indistinguishability, information about encrypted messages is potentially exposed to the adversary in SO-like attack scenarios. This observation calls for a thorough SO analysis of all encryption schemes covered by international standards. The facts that all PKE schemes that so far were formally confirmed to be SO secure require heavy building blocks like lossy trapdoor functions (except for one work discussed in *Previous work*) and that practitioners systematically avoid such building blocks for reasons of efficiency suggest that likely most practical schemes would not withstand SO attacks. Fortunately, however, in this paper we show that virtually all practical PKE constructions provably do meet SO security.

Our approach is complementary to that of prior works: Instead of analysing the asymmetric building blocks of constructions, we observe that SO security is tightly linked to the security of the symmetric building blocks (i.e., symmetric encryption). We particularly show that in the KEM/DEM paradigm for hybrid encryption certain properties of blockcipher-based DEMs suffice to render the overall PKE scheme SO secure (in the ideal cipher model for the blockcipher) *independently of the properties of the KEM*.

In a nutshell, our result is: We introduce a specific property called *simulatability* for blockcipher-based DEMs that is met by virtually all DEMs used in practice and guarantees that if a corresponding DEM is combined with any IND-CCA secure KEM then the overall hybrid PKE scheme achieves SIM-SO-CCA security (in the ideal cipher model). Intuitively, *simulatable* DEMs can be thought of as some form of non-committing encryption in the realm of symmetric cryptography, while non-committing encryption is usually considered in the public-key setting.

Previous work. The SO problem dates back to [12] where the *selective decommitment problem* was studied for commitment schemes. SO notions for encryption first appeared in [3,6]. The first IND-SO-CPA secure encryption scheme in the standard model was given in [3] and is based on lossy encryption (cf. [28]).

Also *deniable encryption* [7] and techniques from *non-committing encryption* [8,21] already allow for constructing SO secure PKE ([11]). Lots of separation and implication results for SO and standard notions were studied in [5,2,6,25]. While it was known that IND-CPA implies *weak*-IND-SO-CPA when messages are drawn pair-wise independently (cf. [12,5]), the implication does not hold for arbitrary (efficiently conditionally resamplable) distributions as recently reported [24]. The result makes use of heavy machinery as *public-coin differing-inputs obfuscation* and *correlation intractable hash functions*. However, IND-CPA implies *weak*-IND-SO-CPA for low-dependency distributions such as Markov chains [19]. Further, SIM-SO secure constructions in the standard model usually (cf. [27]) suffer in efficiency from bit-wise encryption to ensure *efficient openability*. See [23] for current research. SIM-SO-CCA secure PKE schemes are constructed in [18] employing *extended HPS*s and *cross-authentication codes*. This line of research continued in [27] identifying special properties of a KEM, allowing to construct SIM-SO-CCA secure PKE, when combined with *strengthened cross-authentication codes*.

Note that we only consider SO security under sender corruption. Only recently, security under receiver corruption gained some attention [20] while already defined in [1].

Work analysing the SO security of standardised widely-used encryption schemes appeared only recently (in the random oracle model). Concretely, Heuer *et al.* [22] consider Hashed ElGamal encryption (standardised under the name of DHIES) and RSA-OAEP. Unfortunately, the considered versions of these PKE schemes assume messages that are not longer than the output lengths of the used random oracle, i.e., less than 128 bytes. This severely limits the results of [22] for practical considerations.

Paper organization. The structure of this paper is as follows: In the preliminaries (Section 2) we recall basic cryptographic notions, including the definition of SO security. We identify certain combinatorial properties of DEMs, present our main result and highlight the proof ideas in Section 3. Further, we

- show that most widely-used DEMs (in particular those standardised by NIST: CTR, CBC, CCM, and GCM) meet these properties. (Section 4)
- prove that any DEM satisfying standard security notions and these properties in combination with any KEM that meets a standard security notion, results in a SIM-SO-CCA secure PKE scheme in the ideal cipher model. (Section 5)

We conclude in Section 6.

2 Preliminaries

For $n \in \mathbb{N}$ let $[n] := \{1, \ldots, n\}$. We distinguish the following operators for assigning values to variables: We use symbol '$\leftarrow$' when the assigned value results from a constant expression (including the output of a deterministic algorithm), we write '$\leftarrow_U$' when the value is sampled uniformly at random from a finite set, and we write '$\leftarrow_\$$' when the assigned value is the output of a randomised algorithm. If f is a function or a deterministic algorithm that maps elements from a set A to a set B we use notations $f\colon A \to B$ and $A \to f \to B$ interchangeably. If f is a randomised algorithm we correspondingly write $A \to f \to_\$ B$, or simply $f \to_\$ B$ in case the algorithm takes no input. If $A \times B \to f \to C$ is a function then for any $a \in A$ we write $f_a = f(a;\cdot)$ for the partially applied function $B \to f_a \to C;\ b \mapsto f(a,b)$. If R denotes the randomness space of a (randomised) algorithm $A \to f \to_\$ B$, we may write $A \times R \to f \to B$ for its deterministic version. If $A \to f \to B$ is a function or a deterministic algorithm we let $[f] := f(A) \subseteq B$ denote the image of A under f; if $A \to f \to_\$ B$ has randomness space R we correspondingly let $[f] := f(A \times R) \subseteq B$ denote the set of all its possible outputs. When the union $A \cup B$ of two sets A, B is a disjoint union, i.e., if $A \cap B = \emptyset$, we annotate this with $A \mathbin{\dot\cup} B$. For a bitstring x of length at least l we write $\mathrm{msb}_l(x)$ for its left-most l bits and $\mathrm{lsb}_l(x)$ for its right-most l bits ('most/least significant bits').

Our security definitions are based on games played between a challenger and an adversary. These games are expressed using program code and terminate when a 'Stop' command is executed; the argument of the latter is the output of the game. We write $\Pr[G \Rightarrow 1]$ for the probability that game G terminates by running into a 'Stop with 1' instruction.

We next define partial permutations and blockciphers. In our proofs, the former play an important role for the abstraction of the latter.

Definition 1 (Permutation, partial permutation, blockcipher). *For a finite domain $\mathcal{D}$ we denote the set of all permutations on $\mathcal{D}$ with $\mathcal{P}(\mathcal{D})$ and the set of all partial permutations on $\mathcal{D}$ with $\mathcal{PP}(\mathcal{D})$. Precisely, a relation $R \subseteq \mathcal{D} \times \mathcal{D}$ is a partial permutation if $\alpha R \beta, \alpha' R \beta \Rightarrow \alpha = \alpha'$ and $\alpha R \beta, \alpha R \beta' \Rightarrow \beta = \beta'$; relation R is a permutation if in addition $|R| = |\mathcal{D}|$ holds. A blockcipher with key space $\mathcal{K}$ and domain $\mathcal{D}$ is a family $(E_k)_{k\in\mathcal{K}}$ of permutations $E_k \in \mathcal{P}(\mathcal{D})$.*

We associate with a partial permutation $R \in \mathcal{PP}(\mathcal{D})$ the partial functions $\mathcal{D} \to R^+ \to \mathcal{D}$ and $\mathcal{D} \to R^- \to \mathcal{D}$ that evaluate R left-to-right and right-to-left, respectively. For instance, if $(\alpha, \beta) \in R$ then $R^+(\alpha) = \beta$ and $R^-(\beta) = \alpha$. We write $\mathrm{Dom}(R)$ and $\mathrm{Rng}(R)$ for the domain and range of R^+, i.e., for the sets $\{\alpha \in \mathcal{D} \mid \exists \beta : (\alpha, \beta) \in R\}$ and $\{\beta \in \mathcal{D} \mid \exists \alpha : (\alpha, \beta) \in R\}$, respectively. If $\alpha \notin \mathrm{Dom}(R)$ and $\beta \notin \mathrm{Rng}(R)$ we denote with $R \leftarrow R \cup \{(\alpha, \beta)\}$ the operation of 'programming' R such that $R^+(\alpha) = \beta$ and $R^-(\beta) = \alpha$ for the updated R, which is again a partial permutation. Note that any partial permutation can be completed to a (full) permutation by adding sufficiently many such pairs (α, β) to it. More

importantly, if a partial permutation is selected according to the uniform distribution over some subset of $\mathcal{PP}(\mathcal{D})$, it can be extended to a permutation uniformly distributed in $\mathcal{P}(\mathcal{D})$ by adding random such pairs (α, β) to it.

Our definition of keyed hash functions subsumes both message authentication codes and universal hash functions.

Definition 2 (Keyed hash function). *A keyed hash function for a message space $\mathcal{M}$ consists of a key space $\mathcal{K}$, a tag space $\mathcal{T}$, and an efficient function* khf *of the form $\mathcal{K} \times \mathcal{M} \to \text{khf} \to \mathcal{T}$.*

We proceed with specifying the syntax and functionality of DEMs. As a corresponding notion of authenticity we define integrity of ciphertexts [4]. In a nutshell, a DEM offers this feature if no adversary with access to an encapsulation oracle can find a fresh ciphertext that corresponds to a valid message, i.e., is not rejected by the decapsulation algorithm. Relevant in our work is in particular the corresponding one-time notion where the adversary can pose at most one encapsulation query.

Definition 3 (DEM). *A data encapsulation mechanism* (DEM) *for a message space $\mathcal{M}$ consists of a finite key space $\mathcal{K}$, a ciphertext space $\mathcal{C}$, and a pair of efficient algorithms $\mathsf{DEM} = (\mathsf{D.Enc}, \mathsf{D.Dec})$ of the form*

$$\mathcal{K} \times \mathcal{M} \to \mathsf{D.Enc} \to \mathcal{C} \qquad \mathcal{K} \times \mathcal{C} \to \mathsf{D.Dec} \to \mathcal{M} \cup \{\bot\} \ ,$$

where symbol '$\bot$' may be used to indicate errors. Correctness requires that for all $k \in \mathcal{K}$ and $m \in \mathcal{M}$, if $\mathsf{D.Enc}(k, m) = c$ then $\mathsf{D.Dec}(k, c) = m$.

Definition 4 (INT-CTXT secure DEM). *A data encapsulation mechanism is (τ, q_d, ϵ)-OT-INT-CTXT secure if all τ-time adversaries $\mathcal{A}$ that interact in the* OT-INT-CTXT *experiment from Figure 1 and issue at most q_d queries to the* D.Dec *oracle have an advantage of at most ϵ, where we define*

$$\mathbf{Adv}^{\mathsf{OT\text{-}INT\text{-}CTXT}}_{\mathcal{A}} := \Pr[\mathsf{OT\text{-}INT\text{-}CTXT} \Rightarrow 1] \ .$$

This definition can be generalised to $(\tau, q_e, q_d, \epsilon)$-INT-CTXT security by removing line 04 from the experiment and bounding the number of queries to the D.Enc *oracle by q_e.*

Game OT-INT-CTXT	**Oracle** D.Enc(m)	**Oracle** D.Dec(c)
00 $C \leftarrow \emptyset$	04 If $\lvert C\rvert > 0$: Abort	08 If $c \in C$: Abort
01 $k \leftarrow_U \mathcal{K}$	05 $c \leftarrow \mathsf{D.Enc}(k, m)$	09 $m \leftarrow \mathsf{D.Dec}(k, c)$
02 $\mathcal{A}^{\text{D.Enc},\text{D.Dec}}$	06 $C \leftarrow C \cup \{c\}$	10 If $m \neq \bot$:
03 Stop with 0	07 Return c	11 Stop with 1
		12 Return $\bot$

Fig. 1. Security game for defining OT-INT-CTXT security of DEMs. We write 'Abort' as an abbreviation for 'Stop with 0'. Observe that line 04 ensures that the D.Enc oracle is queried at most once.

In most applications a DEM is combined with a KEM to obtain (hybrid) PKE [10]. We recall the concepts of KEMs and PKE below, and include an indistinguishability definition for KEMs.

Definition 5 (KEM). *A key encapsulation mechanism* (KEM) *for a finite key space $\mathcal{K}$ consists of a public-key space $\mathcal{PK}$, a secret-key space $\mathcal{SK}$, a ciphertext space $\mathcal{C}$, and a triple of efficient algorithms $\mathsf{KEM} = (\mathsf{K.Gen}, \mathsf{K.Enc}, \mathsf{K.Dec})$ of the form*

$$\mathsf{K.Gen} \to_{\$} \mathcal{PK} \times \mathcal{SK} \qquad \mathcal{PK} \to \mathsf{K.Enc} \to_{\$} \mathcal{K} \times \mathcal{C} \qquad \mathcal{SK} \times \mathcal{C} \to \mathsf{K.Dec} \to \mathcal{K} \cup \{\bot\} \ ,$$

where symbol '$\bot$' may be used to indicate errors. The randomness space of K.Enc *is typically denoted with $\mathcal{R}$. Correctness requires that for all $(pk, sk) \in [\mathsf{K.Gen}]$, if $(k, c) \in [\mathsf{K.Enc}(pk)]$ then $\mathsf{K.Dec}(sk, c) = k$.*

Definition 6 (IND-CCA secure KEM). *A KEM is (τ, q_d, ϵ)-IND-CCA secure if all τ-time adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that interact in the $\mathsf{IND\text{-}CCA}^b$ experiments from Figure 2 and issue at most q_d queries to the* K.Dec *oracle have an advantage of at most ϵ, where we define*

$$\mathbf{Adv}^{\mathsf{IND\text{-}CCA}}(\mathcal{A}) := \lvert \Pr[\mathsf{IND\text{-}CCA}^0 \Rightarrow 1] - \Pr[\mathsf{IND\text{-}CCA}^1 \Rightarrow 1] \rvert \ .$$

Game IND-CCA^b	**Oracle** K.DEC(c)
00 $C \leftarrow \emptyset$	08 If $c \in C$: Abort
01 $(pk, sk) \leftarrow_\$ \mathsf{K.Gen}$	09 $k \leftarrow \mathsf{K.Dec}(sk, c)$
02 $st \leftarrow_\$ \mathcal{A}_1^{\text{K.DEC}}(pk)$	10 Return k
03 $(k_0^*, c^*) \leftarrow_\$ \mathsf{K.Enc}(pk)$	
04 $k_1^* \leftarrow_U \mathcal{K}$	
05 $C \leftarrow C \cup \{c^*\}$	
06 $b' \leftarrow_\$ \mathcal{A}_2^{\text{K.DEC}}(st, c^*, k_b^*)$	
07 Stop with b'	

Fig. 2. Security games for defining IND-CCA security of KEMs. We write 'Abort' as an abbreviation for 'Stop with 0'.

Definition 7 (PKE). *A scheme for public-key encryption* (PKE) *for a message space $\mathcal{M}$ consists of a public-key space $\mathcal{PK}$, a secret-key space $\mathcal{SK}$, a ciphertext space $\mathcal{C}$, and a triple of efficient algorithms* $\mathsf{PKE} = (\mathsf{P.Gen}, \mathsf{P.Enc}, \mathsf{P.Dec})$ *of the form*

$$\mathsf{P.Gen} \rightarrow_\$ \mathcal{PK} \times \mathcal{SK} \qquad \mathcal{PK} \times \mathcal{M} \rightarrow \mathsf{P.Enc} \rightarrow_\$ \mathcal{C} \qquad \mathcal{SK} \times \mathcal{C} \rightarrow \mathsf{P.Dec} \rightarrow \mathcal{M} \cup \{\perp\} \ ,$$

where symbol '$\perp$' may be used to indicate errors. The randomness space of P.Enc *is typically denoted with $\mathcal{R}$. Correctness requires that for all $(pk, sk) \in [\mathsf{P.Gen}]$ and $m \in \mathcal{M}$, if $c \in [\mathsf{P.Enc}(pk, m)]$ then* $\mathsf{P.Dec}(sk, c) = m$.

Construction 1 (Hybrid encryption) *Take a DEM for a message space $\mathcal{M}$ and a KEM for the key space of the DEM. Then the algorithms in Figure 3 form the hybrid PKE scheme. The randomness space of* P.Enc *coincides with the randomness space of* K.Enc.

Proc P.Gen(r)	**Proc** P.Enc(pk, m, r)	**Proc** P.Dec($sk, \langle c_1, c_2 \rangle$)
00 $(pk, sk) \leftarrow \mathsf{K.Gen}(r)$	02 $(k, c_1) \leftarrow \mathsf{K.Enc}(pk, r)$	05 $k \leftarrow \mathsf{K.Dec}(sk, c_1)$
01 Return (pk, sk)	03 $c_2 \leftarrow \mathsf{D.Enc}(k, m)$	06 If $k = \perp$: Return $\perp$
	04 Return $\langle c_1, c_2 \rangle$	07 $m \leftarrow \mathsf{D.Dec}(k, c_2)$
		08 Return m

Fig. 3. Hybrid construction of PKE from a KEM and a DEM. We write $\langle c_1, c_2 \rangle$ for the encoding of two ciphertext components into one. For clarity we make the randomness used by P.Gen and P.Enc explicit.

We present now the main security definition of this paper: confidentiality under selective opening attacks. Our model is based on works of [6,18] Find a discussion of its details below.

Definition 8 (SIM-SO-CCA secure PKE). *Consider the experiments from Figure 4. For a function $\epsilon\colon \mathbb{N} \rightarrow \mathbb{R}^{\geq 0}$ we say that a PKE scheme is $(\tau, \tau', q_d, \epsilon)$-SIM-SO-CCA secure if for all τ-time adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that interact in the* r-SO-CCA *experiment and issue at most q_d decryption queries there exists a (roughly) τ-time simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ that interacts in the* i-SO-CCA *experiment such that for all τ'-time predicates $\{0,1\}^* \rightarrow \mathsf{Pred} \rightarrow_\$ \{0,1\}$ and all $n \in \mathbb{N}$ the advantage $\mathbf{Adv}^{\mathsf{SIM\text{-}SO\text{-}CCA}}_{\mathcal{A},\mathcal{S},\mathsf{Pred}}(n)$ is at most $\epsilon(n)$, where we define*

$$\mathbf{Adv}^{\mathsf{SIM\text{-}SO\text{-}CCA}}_{\mathcal{A},\mathcal{S},\mathsf{Pred}}(n) := \left|\Pr[\mathsf{r\text{-}SO\text{-}CCA}^{\mathcal{A}}_n \Rightarrow 1] - \Pr[\mathsf{i\text{-}SO\text{-}CCA}^{\mathcal{S}}_n \Rightarrow 1]\right| \ .$$

We give rationale on this formalisation of SO security. The notion compares the information an adversary can deduce about a set of challenge messages in two settings: a real setting (game r-SO-CCA) and an idealised setting (game i-SO-CCA). The real experiment starts with the generation of a key pair. The adversary receives the public key and specifies a message distribution, represented by a randomised circuit $\mathfrak{D}$. Messages $m_1, \ldots, m_n$ are sampled according to this distribution and encrypted using fresh randomnesses $r_1, \ldots, r_n$, and the ciphertexts are given to the adversary which derives some information *out* about the hidden messages. The adversary is supported by two oracles: one that decrypts

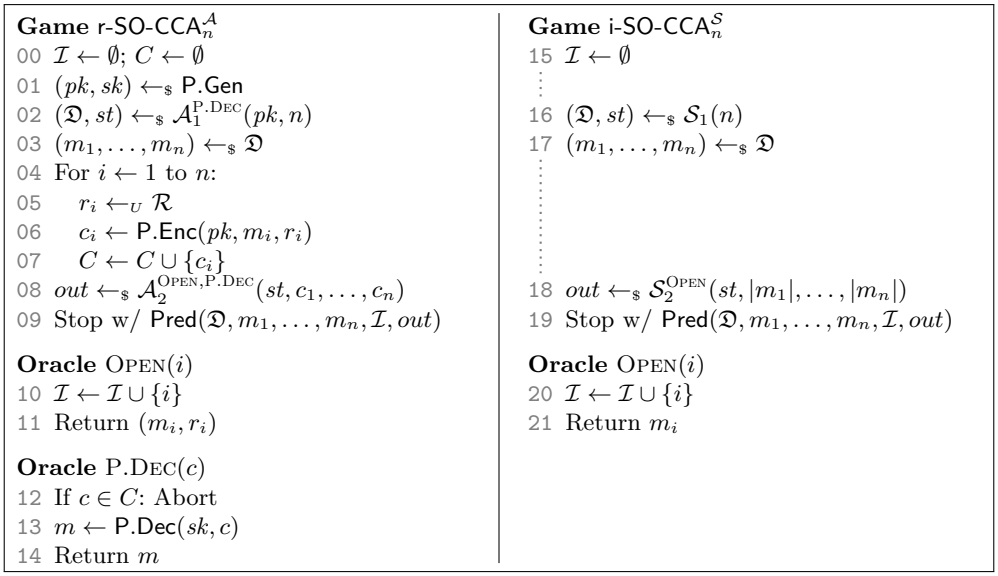

Fig. 4. Security experiments for defining SIM-SO-CCA security of PKE. With $\mathfrak{D}$ we denote a randomised circuit that induces a distribution over $\mathcal{M}^n$. The randomness space of P.Enc is denoted with $\mathcal{R}$. Oracle Open may be called for all $i \in [n]$. We write 'Abort' as an abbreviation for 'Stop with 0'. We show the lines of i-SO-CCA aligned to the ones of r-SO-CCA for easier comparison.

arbitrary ciphertexts and one that opens honest ciphertexts by revealing the corresponding message and the randomness used to encrypt it (this is meant to model sender corruption).

The ideal experiment is similar but with all the artifacts of public key encryption removed: there is no key generation, no ciphertext generation, and no decryption oracle. Beyond that, the adversary (in this context called 'simulator') performs as above: it specifies a message distribution, adaptively requests openings, and derives some information *out* about unopened messages.

Clearly, in the ideal setting the confidentiality of unopened messages is granted (only their lengths leak in line 18, but this is unavoidable for any practical PKE scheme and implicitly also happens in line 08). We thus deem a public key encryption scheme secure under selective opening attacks if the adversary in the real setting cannot draw more conclusions about unopened messages than can be drawn in the ideal setting. Formally, it is required that for every $\mathcal{A}$ for r-SO-CCA there exists a corresponding $\mathcal{S}$ for i-SO-CCA that derives the same information. This is tested by distinguishing predicate Pred, which also takes further environmental information into account, for instance the recorded opening history $\mathcal{I}$. We proceed with some remarks on the model.

In prior works that give simulation-based definitions of SO security there does not seem to be concensus on the order of quantification of $\mathcal{S}$ and Pred. While most papers (cf. [22,27]) allow for the simulator to depend on the distinguishing predicate, the work of [6] implicitly defines a stronger notion that requires the existence of a simulator that is universal. (Interestingly, many papers that exclusively consider the weaker notion actually *do* construct universal simulators.) We adopt the stronger notion and require the simulator to work for any distinguisher.

In the upcoming sections we construct several PKE schemes that are secure under selective opening attacks. The corresponding proofs will idealise a central building block of the schemes, concretely a blockcipher. By consequence, ideal-cipher oracles have to be added to Figure 4. There are various options how and where to do this: It is clear that adversary $\mathcal{A}$ should have access to the ideal cipher, but what about $\mathcal{S}$, what about Pred, and what about $\mathfrak{D}$? It seems that each configuration somehow makes sense and gives rise to an individual variant of SIM-SO-CCA security.[1] Each such notion might have particular strengths and weaknesses, so declaring any of them right or wrong is arbitrary. Ultimately, when proving

[1] A similar situation emerges with NIZK proofs in the random oracle model: In the corresponding ZK definition, shall the distinguisher have access to the random oracle or not? See [30] for a formal treatment and a comparison of the many possible notions.

the SO security of our schemes, we decided to go for a model where, besides the relevant algorithms of the encryption scheme itself, only adversary $\mathcal{A}$ gets access to the ideal cipher.

Notions of SO security under active Attacks. As mentioned in the introduction, three notions for SO security under active attacks exist: {*weak*-IND, *full*-IND, SIM}-SO-CCA. Non of them has emerged as a de-facto standard notion, yet. Clearly, *weak*-IND-SO-CCA suffers from the unnatural restriction to efficiently conditionally resamplable message distributions and security implications for practical applications are unclear. While *full*-IND-SO-CCA would provide security for arbitrary underlying message distributions, as of today, no even a *full*-IND-SO-CPA secure scheme is known.[2]

We note that SIM-SO-CCA — capturing semantic security — does not suffer from any of the above disadvantages (there is no resampling involved) and seems to offer a strong security guarantee.

Little to no research has been conducted relating the SO-CCA notions; [25] shows that IND-CCA does not imply *weak*-IND-CCA in general.

3 Simulatable DEMs and our Main Result

In this section we present our main result on hybrid public key encryption. We define a combinatorial property of a DEM called *simulatability* and show that any KEM and any DEM satisfying standard security notions, if the DEM is in addition simulatable, when composed yield a SIM-SO-CCA secure PKE, in the ideal cipher model [9,17,26].

3.1 Simulatable DEMs

Many practical DEMs are constructed from blockciphers, possibly in combination with further symmetric building blocks like universal hash functions or MACs. We formalise next what it means for a DEM to make use of a blockcipher in a black-box way. Virtually all blockcipher-based DEMs, and in particular those specified by the major standardisation bodies, are of this type. In our definition, $\mathcal{K}$ denotes the key space of the blockcipher and $\mathcal{K}'$ denotes the cartesian product of the key spaces of the remaining cryptographic primitives used by the scheme. For instance, in an encrypt-then-MAC construction, $\mathcal{K}'$ would be the key space of the message authentication code; if the construction requires no further keyed primitive, $\mathcal{K}'$ would be the trivial set containing a single element.

Recall from Definition 1 that $\mathcal{P}(\mathcal{D})$ and $\mathcal{PP}(\mathcal{D})$ denote the sets of all permutations and partial permutations, respectively, on domain $\mathcal{D}$.

Definition 9 (Oracle DEM). *An oracle data encapsulation mechanism*(oDEM) *for a domain $\mathcal{D}$ and a message space $\mathcal{M}$ consists of a finite key space $\mathcal{K}'$, a ciphertext space $\mathcal{C}$, and efficient algorithms* $\mathsf{O.Enc}$ *and* $\mathsf{O.Dec}$ *that have oracle access to a permutation on $\mathcal{D}$ (in both directions) and are of the form*

$$\mathcal{K}' \times \mathcal{M} \to \mathsf{O.Enc}^{\pi} \to \mathcal{C} \qquad \mathcal{K}' \times \mathcal{C} \to \mathsf{O.Dec}^{\pi} \to \mathcal{M} \cup \{\perp\} \ ,$$

where symbol '$\perp$' may be used to indicate errors. Correctness requires that for all $\pi \in \mathcal{P}(\mathcal{D})$, $k' \in \mathcal{K}'$, and $m \in \mathcal{M}$, if $\mathsf{O.Enc}^{\pi}(k', m) = c$ then $\mathsf{O.Dec}^{\pi}(k', c) = m$.

Definition 10 (Permutation-driven DEM). *A DEM for message space $\mathcal{M}$ with keyspace $\mathcal{K}'' = \mathcal{K} \times \mathcal{K}'$ is ($\mathcal{K}$, $\mathcal{D}$)-permutation-driven if there exists an oracle DEM for $\mathcal{D}$ and $\mathcal{M}$ with algorithms $\mathcal{K}' \times \mathcal{M} \to \mathsf{O.Enc}^{\pi} \to \mathcal{C}$ and $\mathcal{K}' \times \mathcal{C} \to \mathsf{O.Dec}^{\pi} \to \mathcal{M} \cup \{\perp\}$ and a blockcipher $(E_k)_{k \in \mathcal{K}}$ on domain $\mathcal{D}$ such that for all $k' \in \mathcal{K}'$ and $m \in \mathcal{M}$ and $c \in \mathcal{C}$ we have*

$$\mathsf{D.Enc}((k, k'), m) = \mathsf{O.Enc}^{E_k}(k', m) \quad \text{and} \quad \mathsf{D.Dec}((k, k'), c) = \mathsf{O.Dec}^{E_k}(k', c) \ . \tag{1}$$

According to this definition, for any specific permutation-driven DEM multiple corresponding oracle DEMs, i.e., $\mathsf{O.Enc}$ and $\mathsf{O.Dec}$ algorithms, might exist. In practice, however, a single canonic specification of these algorithms will stick out. This holds, as we will see, in particular for the standardised DEMs

[2] Note that resampling from any message distribution might require solving some computationally hard problem as reported in [6].

studied in Section 4. For the sake of a concise notation, in this paper we thus assume that suitable $\mathsf{O.Enc}$ and $\mathsf{O.Dec}$ algorithms are always uniquely given.

We next define a combinatorial property called *simulatability* that holds for an oracle DEM if, in principle, the encapsulation algorithm could commit to a ciphertext before seeing the corresponding message; intuitively, this is only possible if the permutation in the oracle is 'flexible enough', i.e., can be 'programmed'. We formalise this idea by splitting the encapsulation routine into two components, Fake and Make. First Fake outputs a ciphertext c without seeing the message m (but it does see the length of m), then Make, on input m, is meant to find a possible (partial) permutation instance $\tilde{\pi}$ under which indeed m would be encapsulated to c. To be useful in our later selective opening related proofs where we want to embed $\tilde{\pi}$ into an ideal cipher, $\tilde{\pi}$ is further required to be uniformly distributed (conditioned on the formulated requirements).

Definition 11 (Simulatable oracle DEM). *Consider an oracle DEM for a domain $\mathcal{D}$ and a message space $\mathcal{M}$ that has an encapsulation algorithm of the form $\mathcal{K}' \times \mathcal{M} \to \mathsf{O.Enc}^{\pi} \to \mathcal{C}$. Consider algorithms* Fake *and* Make *of the form*

$$\mathcal{K}' \times \mathbb{N} \to \mathsf{Fake} \to_{\$} \mathcal{C} \times \Sigma \quad \textit{and} \quad \Sigma \times \mathcal{M} \to \mathsf{Make} \to_{\$} \mathcal{PP}(\mathcal{D}) \ ,$$

where Σ is a state space shared between the two algorithms. We say that the oracle DEM is ϵ-simulatable (by Fake *and* Make*) if for all $k' \in \mathcal{K}'$ and $m \in \mathcal{M}$, for the random variable (defined over the coins of* Fake *and* Make*)*

$$\Pi^m_{k'} = \{\tilde{\pi} : (c, st) \leftarrow_{\$} \mathsf{Fake}(k', |m|); \tilde{\pi} \leftarrow_{\$} \mathsf{Make}(st, m)\}$$

we have

(1) *partial permutation $\Pi^m_{k'}$ can be extended to a uniformly distributed permutation on $\mathcal{D}$, i.e., by 'filling up' $\Pi^m_{k'}$ with random pairs one obtains a permutation uniformly distributed in $\mathcal{P}(\mathcal{D})$;*
(2) *the ciphertext output by* Fake *deviates from the one that would be output by* $\mathsf{O.Enc}$ *if invoked with an extension of the partial permutation output by* Make *with probability at most ϵ. More precisely, for any uniformly distributed extension $\pi \in \mathcal{P}(\mathcal{D})$ of $\Pi^m_{k'}$ we have $\Pr[c \neq \mathsf{O.Enc}^{\pi}(k', m)] \leq \epsilon$ (where the probability is also taken over the random extension of $\Pi^m_{k'}$ to π);*
(3) *the joint running time of $\mathsf{Fake}(k', |m|)$ and $\mathsf{Make}(st, m)$ does not exceed the running time of $\mathsf{O.Enc}(k', m)$, not counting the latter's oracle queries.*

In informal discussions, when we say that a data encapsulation mechanism is simulatable we mean that it is permutation-driven and $\mathsf{Fake}, \mathsf{Make}$ *algorithms exist for which it is ϵ-simulatable with a negligibly small value ϵ.*

Concerning the above definition it is important to understand that the random coins of Fake and Make, and the coins used to extend the partial permutation in items (1) and (2), belong to the same probability space. See Appendix A for an equivalent yet more verbose definition that makes this aspect more explicit.

In line with a comment made above, for all practical DEMs that are simulatable, corresponding specifications for the Fake and Make algorithms emerge canonically. For the sake of notational clarity, from now on we thus assume uniqueness.

Proving Simulatability. We discuss a general technique for proving the simulatability of an oracle DEM. The Fake and Make algorithms are typically explicitly provided in the proof. Fake's strategy is to mimic the behaviour of $\mathsf{O.Enc}$ by executing it and answering blockcipher queries with random elements from $\mathcal{D}$. Make constructs a partial permutation $\tilde{\pi}$ that fits this random assignment by starting with the empty relation $\tilde{\pi} = \emptyset$ and iteratively adding pairs $(\alpha, \beta) \in \mathcal{D} \times \mathcal{D}$ to $\tilde{\pi}$ that help meeting the $\mathsf{O.Enc}^{\tilde{\pi}}(k', m) = c$ goal, always taking care that also the $\alpha\tilde{\pi}\beta, \alpha'\tilde{\pi}\beta \Rightarrow \alpha = \alpha'$ and $\alpha\tilde{\pi}\beta, \alpha\tilde{\pi}\beta' \Rightarrow \beta = \beta'$ requirements from Definition 1 are not violated (Make aborts if simultaneously reaching these conditions turns out to be impossible). Simulatability requirement (1) is achieved by ensuring that for each addition of (α, β) to $\tilde{\pi}$ either α or β are uniformly distributed, conditioned on the prior state of $\tilde{\pi}$. Proving the bound from condition (2) typically requires a combinatorial argument that assesses the probability of collisions. Requirement (3) follows by inspection of the specifications of Fake and Make.

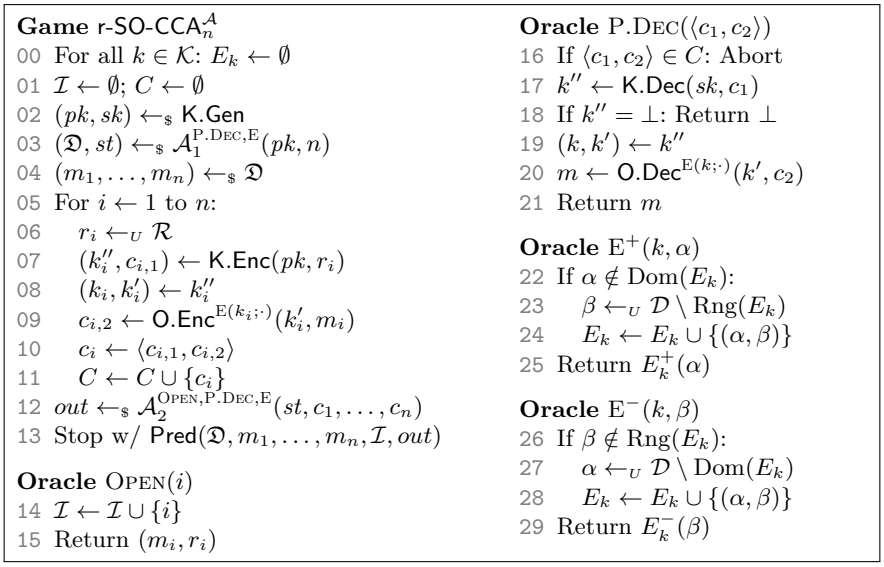

Fig. 5. Game r-SO-CCA adapted towards the analysis of a PKE scheme constructed following the KEM/DEM paradigm using a permutation-driven DEM with corresponding oracle DEM algorithms O.Enc and O.Dec, in the ideal cipher model. We write 'Abort' as an abbreviation for 'Stop with 0'. We further abbreviate the pair E^+, E^- of ideal cipher oracles with just E.

3.2 Selective Opening Security from Simulatable DEMs

Our main result is on the SO security of public-key encryption obtained by combining an arbitrary KEM with a permutation-driven DEM. Our analysis is conducted in the ideal cipher model for the blockcipher underlying the DEM. We give an informal version of our main theorem and an outline of the proof. We caution that some technical preconditions are omitted in the statement as we give it here. See Section 5 for the full theorem statement and proof.

Theorem 1 (informal). *Combine any KEM and any permutation-driven DEM to obtain a PKE scheme. If the KEM is IND-CCA secure, the DEM is OT-INT-CTXT secure and the corresponding oracle DEM is simulatable, then the combined PKE scheme is SIM-SO-CCA secure, in the ideal cipher model.*

We proceed with the proof outline. The goal is to show that for every adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ for the r-SO-CCA game there exists a simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ for the i-SO-CCA game that deduces the same information. In Figure 5 we reproduce the r-SO-CCA game from Figure 4 with the hybrid construction of the encryption scheme, the oracle DEM underlying the DEM, and the ideal cipher model made explicit. (In the i-SO-CCA game there is nothing to be adapted.) We correspondingly equip adversary $\mathcal{A}$ and the DEM algorithms with oracles E^+ and E^- that implement an ideal blockcipher on domain $\mathcal{D}$. In particular, for each key k, oracles $\text{E}^+(k;\cdot)$ and $\text{E}^-(k;\cdot)$ are inverses of each other. For a concise notation, we typically just write E for the pair consisting of E^+ and E^-. We implement ideal cipher E via lazy sampling and keep track of made assignments using a game internal family $(E_k)_{k \in \mathcal{K}}$ of partial permutations $E_k \in \mathcal{PP}(\mathcal{D})$. Note that we do not also provide the KEM algorithms with access to E, meaning we assume the KEM does not use the same blockcipher as the DEM. See Section 5 for a discussion.

When it comes to constructing $\mathcal{S}$ from $\mathcal{A}$, the strategy is to let the former run the latter as a subroutine: Simulator $\mathcal{S}$ converts the own input to an input for $\mathcal{A}$, uses the output of $\mathcal{A}$ as the own output, and answers, and in some cases relays, oracle queries posed by $\mathcal{A}$. We give the footprint of a universal such simulator that leverages on the simulatability of the (permutation-driven) DEM in Figure 6. For the sake of clarity, we simplified the specifications of algorithms $\mathcal{S}_1$ and $\mathcal{S}_2$ quite a bit, removing many technicalities. While we briefly discuss the missing parts below, for the full details of the simulator and a formal analysis we refer to Section 5.

We walk the reader through the design principles of our simulator. What above we refered to as 'deduces the same information' formally requires that the inputs $\mathfrak{D}, m_1, \ldots, m_n, \mathcal{I}, out$ of the Pred invocations in the r-SO-CCA and i-SO-CCA games be similar. This is achieved by letting $\mathcal{S}$ simulate for $\mathcal{A}$ the

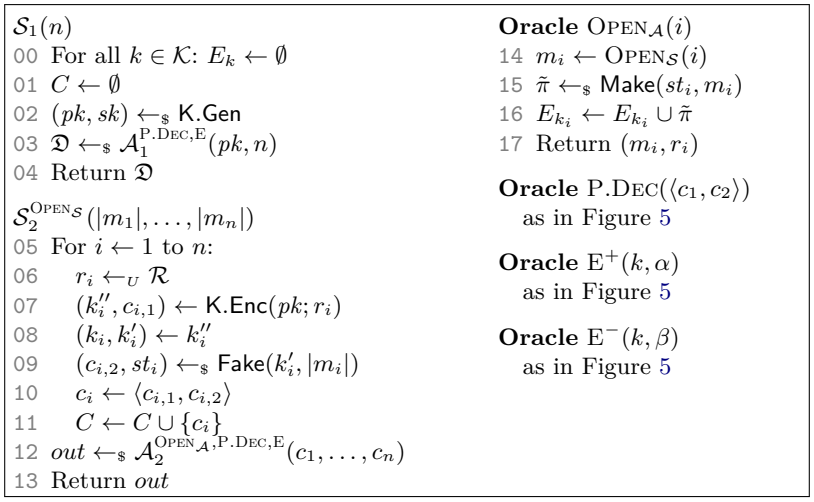

Fig. 6. Simplified version of simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$, constructed from adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. We write $\text{OPEN}_\mathcal{S}$ and $\text{OPEN}_\mathcal{A}$ for the opening oracles provided to $\mathcal{S}_2$ and $\mathcal{A}_2$, respectively. For simplicity we do not annotate the state information passed from $\mathcal{A}_1$ to $\mathcal{A}_2$ and from $\mathcal{S}_1$ to $\mathcal{S}_2$.

environment of r-SO-CCA in a way such that: $\mathcal{S}_1$ forwards the message distribution $\mathfrak{D}$ obtained from $\mathcal{A}_1$ without modification (this also ensures that the distributions of $m_1, \ldots, m_n$ match), $\mathcal{S}_2$ keeps the index sets $\mathcal{I}$ corresponding to $\mathcal{A}_2$'s and its own OPEN queries consistent (by forwarding the queries), and $\mathcal{S}_2$ forwards $\mathcal{A}_2$'s output *out* without modification. The lines in Figure 6 corresponding to these steps are 03,04 and 14 and 12,13, respectively.

Running $\mathcal{A}$ as a subroutine leads to useful results only if $\mathcal{A}$ is exposed to an r-SO-CCA-like environment. Effectively this means that $\mathcal{S}$ has to 'fill all the blank lines' of the i-SO-CCA game in Figure 4. Concretely this involves (a) generating and providing a public key for $\mathcal{A}_1$, (b) providing ciphertexts to $\mathcal{A}_2$ that correspond to messages $m_1, \ldots, m_n$, (c) providing adequate randomness when processing opening queries of $\mathcal{A}_2$, and (d) handling decryption queries of $\mathcal{A}_1$ and $\mathcal{A}_2$. Further, ideal cipher queries of $\mathcal{A}_1$ and $\mathcal{A}_2$ have to be taken care of. The latter is straight-forward when deploying lazy sampling, i.e., using the mechanisms of the r-SO-CCA version from Figure 5. Also (a) and (d) are easy to deal with: The public key *pk* provided to $\mathcal{A}_1$ is a regular KEM key generated by $\mathcal{S}_1$ (lines 02,03); in particular, secret key *sk* is known to $\mathcal{S}$ and can be used to process decryption queries. Concerning (b), creating ciphertexts $c_1, \ldots, c_n$ for $\mathcal{A}_2$ consists, in principle, of two parts: letting the KEM establish session keys and encapsulating messages with the DEM. Component $\mathcal{S}_2$ of our simulator does the former according to the specification, i.e., by invoking algorithm K.Enc with fresh randomness (lines 06,07), while for the latter, as it cannot invoke D.Enc (or, more precisely, O.Enc) for not knowing the messages it needs to encapsulate, it leverages on the simulatability of the DEM and obtains the corresonding ciphertext from an execution of the Fake algorithm (line 09). How $\mathcal{S}_2$ deals with (c) is now immediate: for each created ciphertext it knows the randomness used, so it can release it in an opening query (line 17). Note, however, that knowledge of this randomness brings $\mathcal{A}_2$ into the position to verify the DEM ciphertext components generated by Fake (e.g., by decapsulating or re-encapsulating them); correspondingly, the OPEN oracle in addition runs the Make algorithm and embeds the partial permutation proposed by it into ideal cipher E (lines 15,16). By the definition of simulatability of a DEM, this fixes the ideal cipher such that overall consistency is established.

As announced earlier, in Figure 6 we leave out some details of our simulator. These are related to situations in which $\mathcal{S}$ cannot uphold a proper environment for $\mathcal{A}$ and has to abort its execution. This is the case when Fake and Make fail to properly simulate O.Enc (the definition of simulatability considers a small probability of failure), or if the partial permutation output by Make cannot be embedded into the ideal cipher (line 16). The latter condition can result from various actions of adversary $\mathcal{A}$, for instance (explicitly) from queries to the E oracles, or (implicitly) from evaluations of E during the processing of a decryption query. In the full proof given in Section 5 we show that if the KEM is IND-CCA secure and the DEM is OT-INT-CTXT secure, then the probability is small that any of these conditions is met.

(Very briefly speaking, we use the KEM notion for bounding the probability of explicit queries, and we use the DEM notion for bounding the probability of implicit ones.)

4 Simulatability of practical DEMs

We prove that all blockcipher-based DEMs that were standardised by NIST are permutation-driven and simulatable. Concretely we analyse the CTR and CBC modes of operation (SP 800-38A [13]), a CBC variant with ciphertext stealing (CTS) (Addendum to SP 800-38A [16]), the CCM mode (SP 800-38C [14]), and the GCM mode (SP 800-38D [15]). More precisely, as for our results on selective opening security only those DEMs are relevant that offer ciphertext integrity (cf. Definition 4), instead of plain CTR, CBC, and CBC/CTS encryption we actually analyse their encrypt-then-MAC variants, where we assume arbitrary strongly unforgeable MACs. Further, as CCM and GCM are authenticated encryption schemes with associated data (AEAD [29]), we turn them into DEMs by using them with a fixed nonce N_0 and an empty associated data string A_0.

As the four named modes follow different design principles, some of which might be incompatible with simulatability, analysing all of them is more than just a matter of diligence. For instance, GCM is an encrypt-then-MAC and CCM is a MAC-then-encrypt design. Further, while CTR mode encrypts by XORing blockcipher outputs into the message, CBC mode encrypts by pushing message blocks through the cipher, and CCM combines both approaches.

In the following we specify the mentioned DEMs in their oracle DEM form, assuming that the underlying blockcipher $(E_k)_{k\in\mathcal{K}}$ is over domain $\mathcal{D} = \{0,1\}^\ell$. We show their simulatability by proposing and analysing corresponding Fake and Make algorithms, following the general strategy suggested at the end of Section 3.1.

4.1 CTR-then-MAC

We analyse the DEM obtained by first encrypting the provided message with the CTR0 mode of operation of a blockcipher (counter mode with fixed initial counter value) and then appending a deterministic MAC tag to the ciphertext.

We specify the $\mathsf{O.Enc}$ and $\mathsf{O.Dec}$ algorithms of CTR0-DEM in Figure 7, where we assume that $G\colon [1\,..\,V] \to \mathcal{D}$ denotes a fixed injective function (a 'counter generator') for some sufficiently large value V. The MAC is represented by a keyed hash function $\mathcal{K}' \times \{0,1\}^* \to \mathrm{khf} \to \{0,1\}^T$. The message space of CTR0-DEM is $\mathcal{M} = \{0,1\}^*$ and the ciphertext space is $\mathcal{C} = \{0,1\}^{\geq T}$.

$\mathsf{O.Enc}^\pi(k', m)$	$\mathsf{O.Dec}^\pi(k', c)$
00 Write $\lvert m\rvert$ as $(l-1)\ell + l^*$	12 If $\lvert c\rvert < T$: Return $\perp$
01 Split m into $m_1 \dots m_{l-1} m_l^*$	13 Split c into $\bar{c}t$
02 $m_l \leftarrow m_l^* \,\|\, 0^{\ell - l^*}$	14 If $t \neq \mathrm{khf}(k', \bar{c})$:
03 For $i \leftarrow 1$ to l:	15 Return $\perp$
04 $u_i \leftarrow G(i)$	16 Write $\lvert\bar{c}\rvert$ as $(l-1)\ell + l^*$
05 $v_i \leftarrow \pi(u_i)$	17 Split $\bar{c}$ into $c_1 \dots c_{l-1} c_l^*$
06 $c_i \leftarrow m_i \oplus v_i$	18 $c_l \leftarrow c_l^* \,\|\, 0^{\ell - l^*}$
07 $c_l^* \leftarrow \mathrm{msb}_{l^*}(c_l)$	19 For $i \leftarrow 1$ to l:
08 $\bar{c} \leftarrow c_1 \dots c_{l-1} c_l^*$	20 $u_i \leftarrow G(i)$
09 $t \leftarrow \mathrm{khf}(k', \bar{c})$	21 $v_i \leftarrow \pi(u_i)$
10 $c \leftarrow \bar{c}t$	22 $m_i \leftarrow c_i \oplus v_i$
11 Return c	23 $m_l^* \leftarrow \mathrm{msb}_{l^*}(m_l)$
	24 $m \leftarrow m_1 \dots m_{l-1} m_l^*$
	25 Return m

Fig. 7. CTR0-DEM. Lines 00 and 16 uniquely identify quantities l and l^* such that $l \in \mathbb{N}^{\geq 1}$ and $0 \leq l^* < \ell$, and $\lvert m\rvert = (l-1)\ell + l^*$ and $\lvert\bar{c}\rvert = (l-1)\ell + l^*$, respectively. Correspondingly, line 01 assumes $\lvert m_1\rvert = \ldots = \lvert m_{l-1}\rvert = \ell$ and $\lvert m_l^*\rvert = l^*$, and line 17 assumes $\lvert c_1\rvert = \ldots = \lvert c_{l-1}\rvert = \ell$ and $\lvert c_l^*\rvert = l^*$. Further, line 13 assumes $\lvert t\rvert = T$.

Lemma 1. *CTR0-DEM is ϵ-simulatable with $\epsilon = (\lceil L/\ell \rceil^2 - \lceil L/\ell \rceil)/2^{\ell+1}$, where L is the maximum message length (in bits).*

Proof. Consider algorithms Fake and Make from Figure 8. The idea of Fake is to compute intermediate ciphertext $\bar{c}$ on basis of uniformly distributed blockcipher outputs (see how line 01 of Fake replaces l-many iterations of line 06 of O.Enc), but to compute the MAC tag on $\bar{c}$ faithfully. Note that the correct length of $\bar{c}$ is known to Fake as it coincides with the length of m. Inspection shows that, given m, algorithm Make finds a minimal partial permutation $\tilde{\pi}$ such that Fake and Make jointly mimic the behaviour of O.Enc (see here how lines 15–18 of Make arrange the entries of $\tilde{\pi}$ such that they are consistent with lines 05–06 of O.Enc). In some invocations of the algorithms, the described process might fail (lines 16, 17), namely when partial permutation $\tilde{\pi}$ would become inconsistent (i.e., the updated $\tilde{\pi}$ would stop being an element of $\mathcal{PP}$). In such cases Make aborts, outputting the empty partial permutation $\tilde{\pi} = \emptyset$.

We next show that the conditions from Definition 11 are met. Observe that, as Fake picks values $c_1, \ldots, c_l$ uniformly and independently of each other, the same holds for the values $v_1, \ldots, v_l$ computed in line 15. That is, in each iteration of line 18 a value v_i is added to $\mathrm{Rng}(\tilde{\pi})$ that is uniform conditioned on the then current state of $\mathrm{Rng}(\tilde{\pi})$. Thus condition (1) holds. To establish the correctness bound of condition (2) we analyse the probability that Make aborts. By the injectivity of function G the u_i-values from line 14 are pairwise distinct, so the abort condition of line 16 is never met. Further, as values v_i computed in line 15 are uniformly distributed and independent of each other, the abort condition of line 17 is met with probability $\epsilon = (0 + \ldots + (l-1))/|\mathcal{D}| = ((l^2 - l)/2)/|\mathcal{D}|$ (accumulated over all iterations of the loop). Plugging in the maximum value $l = \lceil L/\ell \rceil$ gives the bound claimed in the statement. Condition (3) is clear. □

Fake$(k', \lvert m \rvert)$	Make(st, m)
00 Write $\lvert m \rvert$ as $(l-1)\ell + l^*$	08 $\tilde{\pi} \leftarrow \emptyset$
01 $c_1, \ldots, c_l \leftarrow_U \mathcal{D}$	09 Write $\lvert m \rvert$ as $(l-1)\ell + l^*$
02 $c_l^* \leftarrow \mathrm{msb}_{l^*}(c_l)$	10 Parse st as $(c_1, \ldots, c_l)$
03 $\bar{c} \leftarrow c_1 \ldots c_{l-1} c_l^*$	11 Split m into $m_1 \ldots m_{l-1} m_l^*$
04 $t \leftarrow \mathrm{khf}(k', \bar{c})$	12 $m_l \leftarrow m_l^* \,\|\, 0^{\ell - l^*}$
05 $c \leftarrow \bar{c}t$	13 For $i \leftarrow 1$ to l:
06 $st \leftarrow (c_1, \ldots, c_l)$	14 $\quad u_i \leftarrow G(i)$
07 Return c, st	15 $\quad v_i \leftarrow m_i \oplus c_i$
	16 $\quad$ If $u_i \in \mathrm{Dom}(\tilde{\pi})$: Abort
	17 $\quad$ If $v_i \in \mathrm{Rng}(\tilde{\pi})$: Abort
	18 $\quad \tilde{\pi} \leftarrow \tilde{\pi} \cup \{(u_i, v_i)\}$
	19 Return $\tilde{\pi}$

Fig. 8. Fake and Make for CTR0-DEM. We write 'Abort' as an abbreviation for 'Return $\emptyset$'.

4.2 CBC-then-MAC

We consider the DEM obtained by encrypting the message with CBC0 mode (cipher block chaining with initialisation vector zero) and appending a MAC tag to the ciphertext. As a variant we also look at CBC0-CTS (CBC0 with 'ciphertext stealing') that supports a complementary message space.

We specify the O.Enc and O.Dec algorithms of CBC-DEM in Figure 9 and of CBC-CTS-DEM in Figure 10. Similarly as for CTR0-DEM, the MAC is represented by a keyed hash function of the form $\mathcal{K}' \times \{0,1\}^* \to \mathrm{khf} \to \{0,1\}^T$. The message space of CBC-DEM consists of all messages that have a length that is a multiple of the blocklength ℓ, i.e., $\mathcal{M} = \bigcup_{\lambda \geq \ell, \ell \mid \lambda} \{0,1\}^\lambda$; the ciphertext space is $\mathcal{C} = \bigcup_{\lambda \geq \ell, \ell \mid \lambda} \{0,1\}^{\lambda + T}$. In contrast, CBC-CTS-DEM supports all message lengths that are not a multiple of ℓ, with a minimum value of $\ell + 1$; formally, $\mathcal{M} = \bigcup_{\lambda \geq \ell, \ell \nmid \lambda} \{0,1\}^\lambda$ and $\mathcal{C} = \bigcup_{\lambda \geq \ell, \ell \nmid \lambda} \{0,1\}^{\lambda + T}$. Together, CBC-DEM and CBC-CTS-DEM can handle messages of any length not smaller than ℓ.[3]

[3] Instead of specifying different algorithms for different classes of message length, one could also join them together to a single, more general algorithm. This is usually done in standards [16], but we abstain from doing so in this document to avoid rather obstructing case distinctions in the analysis.

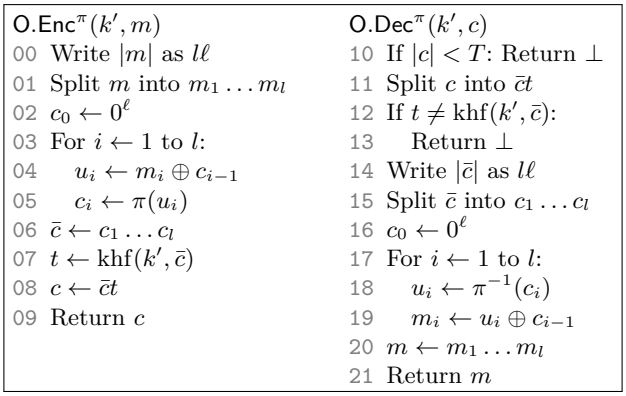

Fig. 9. CBC-DEM (for multi-block messages). Lines 00 and 14 identify quantity $l \in \mathbb{N}^{\geq 0}$ such that $|m| = l\ell$ and $|\bar{c}| = l\ell$, respectively. Correspondingly, line 01 assumes $|m_1| = \ldots = |m_l| = \ell$ and line 15 assumes $|c_1| = \ldots = |c_l| = \ell$. Further, line 11 assumes $|t| = T$.

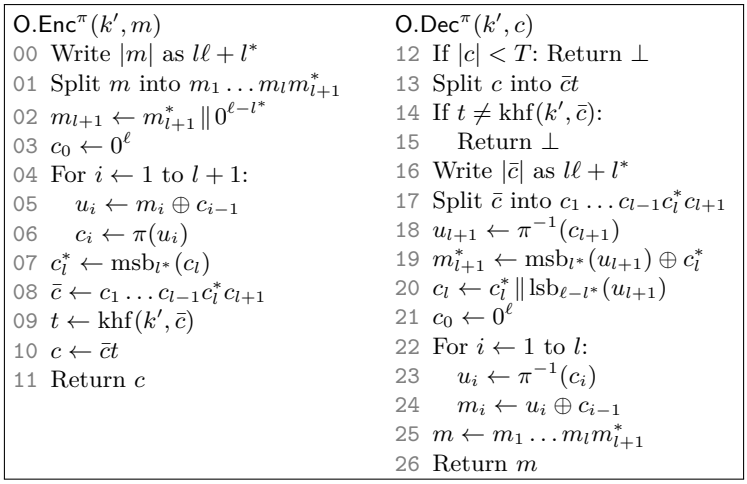

Fig. 10. CBC-CTS-DEM (for messages that require padding). Lines 00 and 16 uniquely identify quantities l and l^* such that $l \in \mathbb{N}^{\geq 1}$ and $1 \leq l^* < \ell$, and $|m| = l\ell + l^*$ and $|\bar{c}| = l\ell + l^*$, respectively. Correspondingly, line 01 assumes $|m_1| = \ldots = |m_l| = \ell$ and $|m^*_{l+1}| = l^*$, and line 17 assumes $|c_1| = \ldots = |c_{l-1}| = \ell$ and $|c^*_l| = l^*$ and $|c_{l+1}| = \ell$. Further, line 13 assumes $|t| = T$.

Lemma 2. *CBC-DEM is ϵ-simulatable where $\epsilon = ((L/\ell)^2 - (L/\ell))/2^\ell$, and CBC-CTS-DEM is ϵ-simulatable with $\epsilon = (\lfloor L/\ell \rfloor^2 + \lfloor L/\ell \rfloor)/2^\ell$, where L is the maximum message length (in bits).*

Proof. The proof is similar to the one of Lemma 1. Consider algorithms Fake and Make from Figure 11. The idea of Fake is to compute intermediate ciphertext $\bar{c}$ on basis of uniformly distributed blockcipher outputs (see how line 01 of Fake replaces l-many iterations of line 05 of O.Enc), but to compute the MAC tag on $\bar{c}$ faithfully. Note that the correct length of $\bar{c}$ is known to Fake as it coincides with the length of m. Inspection shows that, given m, algorithm Make finds a minimal partial permutation $\tilde{\pi}$ such that Fake and Make jointly mimic the behaviour of O.Enc (see here how lines 13–16 of Make arrange the entries of $\tilde{\pi}$ such that they are consistent with lines 04–05 of O.Enc). In some invocations of the algorithms, the described process might fail (lines 14, 15), namely when partial permutation $\tilde{\pi}$ would become inconsistent. In such cases Make aborts, outputting the empty partial permutation $\tilde{\pi} = \emptyset$.

We next show that the conditions from Definition 11 are met. Observe that, as Fake picks values $c_1, \ldots, c_l$ uniformly and independently of each other, in each iteration of line 16 a value c_i is added to $\mathrm{Rng}(\tilde{\pi})$ that is uniform conditioned on the then current state of $\mathrm{Rng}(\tilde{\pi})$. Thus condition (1) holds. To establish the correctness bound of condition (2) we analyse the probability that Make aborts. With values $c_1, \ldots, c_{l-1}$ also the values $u_2, \ldots, u_l$ computed in line 13 are uniformly distributed and independent of each other, so the abort condition of line 14 is met with probability $(0 + \ldots + (l-1))/|\mathcal{D}| = ((l^2 - l)/2)/|\mathcal{D}|$

(accumulated over all iterations of the loop). The same bound holds for line 15. Plugging in the maximum value $l = L/\ell$ gives the bound claimed in the statement. Condition (3) is clear.

Algorithms Fake and Make for CBC-CTS-DEM are given in Figure 12. The analysis is similar. Here, however, we have $l = \lfloor L/\ell \rfloor$ and for lines 16 and 17 the accumulated probabilities of abort amount to $(0 + \ldots + l)/|\mathcal{D}|$ each. □

$\mathsf{Fake}(k', \lvert m\rvert)$	$\mathsf{Make}(st, m)$
00 Write $\lvert m\rvert$ as $l\ell$	07 $\tilde{\pi} \leftarrow \emptyset$
01 $c_1, \ldots, c_l \leftarrow_U \mathcal{D}$	08 Write $\lvert m\rvert$ as $l\ell$
02 $\bar{c} \leftarrow c_1 \ldots c_l$	09 Parse st as $(c_1, \ldots, c_l)$
03 $t \leftarrow \mathrm{khf}(k', \bar{c})$	10 Split m into $m_1 \ldots m_l$
04 $c \leftarrow \bar{c}t$	11 $c_0 \leftarrow 0^\ell$
05 $st \leftarrow (c_1, \ldots, c_l)$	12 For $i \leftarrow 1$ to l:
06 Return c, st	13 $u_i \leftarrow m_i \oplus c_{i-1}$
	14 If $u_i \in \mathrm{Dom}(\tilde{\pi})$: Abort
	15 If $c_i \in \mathrm{Rng}(\tilde{\pi})$: Abort
	16 $\tilde{\pi} \leftarrow \tilde{\pi} \cup \{(u_i, c_i)\}$
	17 Return $\tilde{\pi}$

Fig. 11. Fake and Make for CBC-DEM. We write 'Abort' as an abbreviation for 'Return $\emptyset$'.

$\mathsf{Fake}(k', \lvert m\rvert)$	$\mathsf{Make}(st, m)$
00 Write $\lvert m\rvert$ as $l\ell + l^*$	08 $\tilde{\pi} \leftarrow \emptyset$
01 $c_1, \ldots, c_{l+1} \leftarrow_U \mathcal{D}$	09 Write $\lvert m\rvert$ as $l\ell + l^*$
02 $c_l^* \leftarrow \mathrm{msb}_{l^*}(c_l)$	10 Parse st as $(c_1, \ldots, c_{l+1})$
03 $\bar{c} \leftarrow c_1 \ldots c_{l-1} c_l^* c_{l+1}$	11 Split m into $m_1 \ldots m_l m_{l+1}^*$
04 $t \leftarrow \mathrm{khf}(k', \bar{c})$	12 $m_{l+1} \leftarrow m_{l+1}^* \,\|\, 0^{\ell - l^*}$
05 $c \leftarrow \bar{c}t$	13 $c_0 \leftarrow 0^\ell$
06 $st \leftarrow (c_1, \ldots, c_{l+1})$	14 For $i \leftarrow 1$ to $l+1$:
07 Return c, st	15 $u_i \leftarrow m_i \oplus c_{i-1}$
	16 If $u_i \in \mathrm{Dom}(\tilde{\pi})$: Abort
	17 If $c_i \in \mathrm{Rng}(\tilde{\pi})$: Abort
	18 $\tilde{\pi} \leftarrow \tilde{\pi} \cup \{(u_i, c_i)\}$
	19 Return $\tilde{\pi}$

Fig. 12. Fake and Make for CBC-CTS-DEM. We write 'Abort' as an abbreviation for 'Return $\emptyset$'.

4.3 CCM

We analyse the CCM mode of operation ('CTR mode with CBC-MAC') with fixed nonce and associated data field; we call this mode CCM0-DEM. CCM is parameterised by an authentication tag length T, a formatting function $F\colon \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{D}^+$ (where $\mathcal{N}$ and $\mathcal{A}$ denote the nonce space and the associated data space, respectively), and a counter generation function $G\colon \mathcal{N} \times [0\,..\,V] \to \mathcal{D}$, where V is a sufficiently large value. While only one set of instantiations of F and G is suggested in SP 800-38C (and if it is chosen the resulting version of CCM is the one used in wireless encryption standard IEEE 802.11), the specification is explicitly modular in the sense that it works with any F and G that meet certain conditions. Amongst others, the conditions listed in [14] imply that for all $N \in \mathcal{N}$ the function $G(N; \cdot)$ is injective and that for all $(N, A, m) \in \mathcal{N} \times \mathcal{A} \times \mathcal{M}$ and $z_0 \ldots z_r = F(N, A, m)$ we have that $z_0 \notin G(N, [0\,..\,V])$. Now, if we fix any nonce N_0 and any associated data string A_0 (e.g., the all-zero string for N_0 and the empty string for A_0) and define the restrictions $F_0\colon \mathcal{M} \to \mathcal{D}^+;\ m \mapsto F(N_0, A_0, m)$ and $G_0\colon [0\,..\,V] \to \mathcal{D};\ i \mapsto G(N_0, i)$, then the algorithms of the resulting oracle DEM associated with CCM are given in Figure 13. The message space of CCM0-DEM is $\mathcal{M} = \{0,1\}^*$ and the ciphertext space is $\mathcal{C} = \{0,1\}^{\geq T}$.

$\mathsf{O.Enc}^{\pi}(k', m)$
00 $z_0 \dots z_r \leftarrow F_0(m)$
01 $y_0 \leftarrow \pi(z_0)$
02 For $i \leftarrow 1$ to r:
03 $\quad x_i \leftarrow z_i \oplus y_{i-1}$
04 $\quad y_i \leftarrow \pi(x_i)$
05 $u_0 \leftarrow G_0(0)$
06 $v_0 \leftarrow \pi(u_0)$
07 $t \leftarrow y_r \oplus v_0$
08 $t^* \leftarrow \mathrm{msb}_T(t)$
09 Write $|m|$ as $(l-1)\ell + l^*$
10 Split m into $m_1 \dots m_{l-1} m_l^*$
11 $m_l \leftarrow m_l^* \,\|\, 0^{\ell - l^*}$
12 For $j \leftarrow 1$ to l:
13 $\quad u_j \leftarrow G_0(j)$
14 $\quad v_j \leftarrow \pi(u_j)$
15 $\quad c_j \leftarrow m_j \oplus v_j$
16 $c_l^* \leftarrow \mathrm{msb}_{l^*}(c_l)$
17 $c \leftarrow c_1 \dots c_{l-1} c_l^* t^*$
18 Return c

$\mathsf{O.Dec}^{\pi}(k', c)$
19 If $|c| < T$: Return $\perp$
20 Write $|c|$ as $(l-1)\ell + l^* + T$
21 Split c into $c_1 \dots c_{l-1} c_l^* t^*$
22 $c_l \leftarrow c_l^* \,\|\, 0^{\ell - l^*}$
23 For $j \leftarrow 1$ to l:
24 $\quad u_j \leftarrow G_0(j)$
25 $\quad v_j \leftarrow \pi(u_j)$
26 $\quad m_j \leftarrow c_j \oplus v_j$
27 $m_l^* \leftarrow \mathrm{msb}_{l^*}(m_l)$
28 $m \leftarrow m_1 \dots m_{l-1} m_l^*$
29 $z_0 \dots z_r \leftarrow F_0(m)$
30 $y_0 \leftarrow \pi(z_0)$
31 For $i \leftarrow 1$ to r:
32 $\quad x_i \leftarrow z_i \oplus y_{i-1}$
33 $\quad y_i \leftarrow \pi(x_i)$
34 $u_0 \leftarrow G_0(0)$
35 $v_0 \leftarrow \pi(u_0)$
36 $t \leftarrow y_r \oplus v_0$
37 If $t^* \neq \mathrm{msb}_T(t)$: Return $\perp$
38 Return m

Fig. 13. CCM0-DEM. Lines 09 and 20 uniquely identify quantities l and l^* such that $l \in \mathbb{N}^{\geq 1}$ and $0 \leq l^* < \ell$, and $|m| = (l-1)\ell + l^*$ and $|c| = (l-1)\ell + l^* + T$, respectively. Correspondingly, line 10 assumes $|m_1| = \ldots = |m_{l-1}| = \ell$ and $|m_l^*| = l^*$, and line 21 assumes $|c_1| = \ldots = |c_{l-1}| = \ell$ and $|c_l^*| = l^*$ and $|t^*| = T$.

Lemma 3. *CCM0-DEM is ϵ-simulatable with $\epsilon \leq \lfloor L/\ell \rfloor^2 / 2^{\ell-2}$, where L is the maximum message length (in bits).*

Proof. Consider algorithms Fake and Make from Figure 14. The idea of Fake is to compute the visible ciphertext components on basis of uniformly distributed blockcipher outputs while completely ignoring the blockcipher invocations of CCM's internal CBC-MAC computation (see how line 07 and l-many iterations of line 15 of O.Enc (in Figure 13) are replaced by lines 00 and 03 of Fake, while lines 01 and 04 of O.Enc have no counterpart). Inspection shows that, given m, algorithm Make finds a minimal partial permutation $\tilde{\pi}$ such that Fake and Make jointly mimic the behaviour of O.Enc (see here how lines 24–27, 30–33, 35–38, 43–46 of Make arrange the entries of $\tilde{\pi}$ such that they are consistent with lines 01, 04, 06/07, 14/15 of O.Enc). In some invocations of the algorithms, the described process might fail (in lines 25/26, 31/32, 36/37, 44/45), namely when partial permutation $\tilde{\pi}$ would become inconsistent. In such cases Make aborts, outputting the empty partial permutation $\tilde{\pi} = \emptyset$.

$\mathsf{Fake}(k', |m|)$
00 $t \leftarrow_U \mathcal{D}$
01 $t^* \leftarrow \mathrm{msb}_T(t)$
02 Write $|m|$ as $(l-1)\ell + l^*$
03 $c_1, \dots, c_l \leftarrow_U \mathcal{D}$
04 $c_l^* \leftarrow \mathrm{msb}_{l^*}(c_l)$
05 $c \leftarrow c_1 \dots c_{l-1} c_l^* t^*$
06 $st \leftarrow (t, c_1, \dots, c_l)$
07 Return c, st

$\mathsf{Make}(st, m)$
20 $\tilde{\pi} \leftarrow \emptyset$
21 Write $|m|$ as $(l-1)\ell + l^*$
22 Parse st as $(t, c_1, \dots, c_l)$
23 $z_0 \dots z_r \leftarrow F_0(m)$
24 $y_0 \leftarrow_U \mathcal{D}$
25 If $z_0 \in \mathrm{Dom}(\tilde{\pi})$: Abort
26 If $y_0 \in \mathrm{Rng}(\tilde{\pi})$: Abort
27 $\tilde{\pi} \leftarrow \tilde{\pi} \cup \{(z_0, y_0)\}$
28 For $i \leftarrow 1$ to r:
29 $\quad x_i \leftarrow z_i \oplus y_{i-1}$
30 $\quad y_i \leftarrow_U \mathcal{D}$
31 $\quad$ If $x_i \in \mathrm{Dom}(\tilde{\pi})$: Abort
32 $\quad$ If $y_i \in \mathrm{Rng}(\tilde{\pi})$: Abort
33 $\quad \tilde{\pi} \leftarrow \tilde{\pi} \cup \{(x_i, y_i)\}$
34 $u_0 \leftarrow G_0(0)$
35 $v_0 \leftarrow y_r \oplus t$
36 If $u_0 \in \mathrm{Dom}(\tilde{\pi})$: Abort
37 If $v_0 \in \mathrm{Rng}(\tilde{\pi})$: Abort
38 $\tilde{\pi} \leftarrow \tilde{\pi} \cup \{(u_0, v_0)\}$
39 Split m into $m_1 \dots m_{l-1} m_l^*$
40 $m_l \leftarrow m_l^* \,\|\, 0^{\ell - l^*}$
41 For $j \leftarrow 1$ to l:
42 $\quad u_j \leftarrow G_0(j)$
43 $\quad v_j \leftarrow m_j \oplus c_j$
44 $\quad$ If $u_j \in \mathrm{Dom}(\tilde{\pi})$: Abort
45 $\quad$ If $v_j \in \mathrm{Rng}(\tilde{\pi})$: Abort
46 $\quad \tilde{\pi} \leftarrow \tilde{\pi} \cup \{(u_j, v_j)\}$
47 Return $\tilde{\pi}$

Fig. 14. Fake and Make for CCM0-DEM. We write 'Abort' as an abbreviation for 'Return $\emptyset$'.

We next show that the requirements from Definition 11 are met. To see that condition (1) holds, observe that in Make the values y_0, y_i, v_0, and v_j are uniformly distributed and independent of each other at the point they are added to $\mathrm{Rng}(\tilde{\pi})$ in lines 27, 33, 38, 46. To establish the correctness bound of condition (2) we assess the probability that Make aborts. Using a similar analysis as in the proof of Lemma 1 we obtain the following (accumulated) probabilities: The abort conditions in lines 25 and 26 are never met; for lines 31 and 32 the probabilities are $(1+\ldots+r)/|\mathcal{D}|$ each; by the properties of CCM's functions F_0 and G_0, for lines 36 and 37 the probabilities are $r/|\mathcal{D}|$ and $(r+1)/|\mathcal{D}|$; for line 44 the probability is $lr/|\mathcal{D}|$; finally, for line 45 the probability is $((r+2)+\ldots+(r+l+1))/|\mathcal{D}|$. If we assume reasonable behaviour of function F_0 and let $r=l$, we obtain quantity $4l^2/|\mathcal{D}|$ as an upper bound for the sum of these probabilities. This establishes the claimed bound. Condition (3) is clear. □

4.4 GCM

The GCM mode of operation ('Galois/Counter Mode') is a nonce-based AEAD parameterised by an authentication tag length T. To deploy GCM as a DEM we use it with a fixed nonce and an empty associated data field and call this version GCM0-DEM. Internally, GCM combines CTR mode encryption with a Wegman-Carter MAC. The former uses an injective counter generation function $G\colon [0\,..\,V] \to \mathcal{D}\setminus\{0^\ell\}$, where V is a sufficiently large value, and the latter is built around a polynomial-based universal hash function named GHASH defined over finite field $\mathrm{GF}(2^\ell)$. For our purposes it suffices to represent the MAC by a keyed hash function of the form $\mathcal{D}\times\{0,1\}^* \to \mathrm{khf} \to \mathcal{D}$. The algorithms of GCM0-DEM, in the abstraction of an oracle DEM, are specified in Figure 15. The supported message space is $\mathcal{M}=\{0,1\}^*$, and the ciphertext space is $\mathcal{C}=\{0,1\}^{\geq T}$.

$\mathsf{O.Enc}^\pi(k',m)$	$\mathsf{O.Dec}^\pi(k',c)$
00 Write $\lvert m\rvert$ as $(l-1)\ell+l^*$	18 If $\lvert c\rvert < T$: Return $\perp$
01 Split m into $m_1\ldots m_{l-1}m_l^*$	19 Write $\lvert c\rvert$ as $(l-1)\ell+l^*+T$
02 $m_l \leftarrow m_l^* \,\|\, 0^{\ell-l^*}$	20 Split c into $\bar{c}t^*$
03 For $i \leftarrow 1$ to l:	21 $u \leftarrow 0^\ell$
04 $\quad u_i \leftarrow G(i)$	22 $v \leftarrow \pi(u)$
05 $\quad v_i \leftarrow \pi(u_i)$	23 $h \leftarrow \mathrm{khf}(v,\bar{c})$
06 $\quad c_i \leftarrow m_i \oplus v_i$	24 $u_0 \leftarrow G(0)$
07 $c_l^* \leftarrow \mathrm{msb}_{l^*}(c_l)$	25 $v_0 \leftarrow \pi(u_0)$
08 $\bar{c} \leftarrow c_1\ldots c_{l-1}c_l^*$	26 $t \leftarrow h \oplus v_0$
09 $u \leftarrow 0^\ell$	27 If $t^* \neq \mathrm{msb}_T(t)$: Return $\perp$
10 $v \leftarrow \pi(u)$	28 Split $\bar{c}$ into $c_1\ldots c_{l-1}c_l^*$
11 $h \leftarrow \mathrm{khf}(v,\bar{c})$	29 $c_l \leftarrow c_l^* \,\|\, 0^{\ell-l^*}$
12 $u_0 \leftarrow G(0)$	30 For $i \leftarrow 1$ to l:
13 $v_0 \leftarrow \pi(u_0)$	31 $\quad u_i \leftarrow G(i)$
14 $t \leftarrow h \oplus v_0$	32 $\quad v_i \leftarrow \pi(u_i)$
15 $t^* \leftarrow \mathrm{msb}_T(t)$	33 $\quad m_i \leftarrow c_i \oplus v_i$
16 $c \leftarrow \bar{c}t^*$	34 $m_l^* \leftarrow \mathrm{msb}_{l^*}(m_l)$
17 Return c	35 $m \leftarrow m_1\ldots m_{l-1}m_l^*$
	36 Return m

Fig. 15. GCM0-DEM. Lines 00 and 19 uniquely identify quantities l and l^* such that $l \in \mathbb{N}^{\geq 1}$ and $0 \leq l^* < \ell$, and $|m| = (l-1)\ell+l^*$ and $|c| = (l-1)\ell+l^*+T$, respectively. Correspondingly, line 01 assumes $|m_1| = \ldots = |m_{l-1}| = \ell$ and $|m_l^*| = l^*$, line 20 assumes $|t^*| = T$, and line 28 assumes $|c_1| = \ldots = |c_{l-1}| = \ell$ and $|c_l^*| = l^*$.

Lemma 4. *GCM0-DEM is ϵ-simulatable with $\epsilon \leq (\lceil L/\ell\rceil^2 + 4\lceil L/\ell\rceil)/2^{\ell-1}$, where L is the maximum message length (in bits).*

Proof. The structure of GCM0-DEM is quite similar to the one of CTR0-DEM: both modes first encrypt the message using CTR mode, then they append a MAC tag to the ciphertext. Two potentially interesting differences are that (a) in GCM0-DEM, the MAC key is derived by enciphering the value 0^ℓ under the blockcipher, and (b) in GCM0-DEM, the MAC tag is a GHASH value that is blinded with a blockcipher

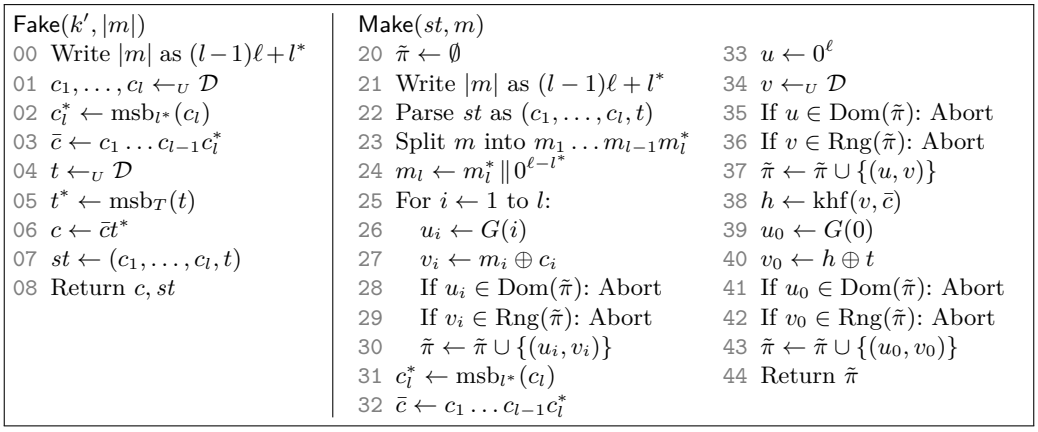

Fig. 16. Fake and Make for GCM0-DEM. We write 'Abort' as an abbreviation for 'Return $\emptyset$'.

output (as is standard for Wegman-Carter MACs). Despite these differences, extending the proof of Lemma 1 to the GCM setting is straight-forward. The corresponding Fake and Make algorithms are given in Figure 16 and do not require further explanation.

We show that the requirements from Definition 11 are met. To see that condition (1) holds, observe that in Make the values v_i, v, and v_0 are uniformly distributed and independent of each other at the point they are added to $\mathrm{Rng}(\tilde{\pi})$ in lines 30, 37, 43. To establish the correctness bound of condition (2) we assess the probability that Make aborts. The analysis is particularly simple: the conditions in lines 28, 35, 41 are never met by construction, and the conditions in lines 29, 36, 42 are met with a total probability of $(0 + \ldots + (l+1))/|\mathcal{D}|$. This establishes the claimed bound. Condition (3) is clear. □

5 A Formal Treatment of Our Main Result

We anticipated the main result of this paper in Section 3: Any (hybrid) PKE scheme constructed from a KEM and a permutation-driven DEM offers SIM-SO-CCA security in the ideal cipher model, if the KEM provides confidentiality (IND-CCA), the DEM provides authenticity (OT-INT-CTXT), and the DEM is simulatable. Prerequisites like IND-CCA and OT-INT-CTXT on the KEM and DEM, respectively, are standard for proofs of the IND-CCA security of hybrid encryption, so the important finding is that the added constraint of simulatability suffices to lift security to the stronger notion of SO security.[4]

We discussed an informal version of our result in Section 3.2. Recall from the included proof sketch that an important subgoal was bounding the probability of the ideal cipher being evaluated on input a key established by the KEM before a corresponding Open query is posed. (If the cipher is evaluated earlier, the partial permutation found by Fake and Make cannot be smoothly embedded into it any more.) In the following we argue that without putting further restrictions on the KEM, bounding this probability to any small value is in general impossible. Indeed, assume for a moment a KEM where K.Enc, before outputting a key k and a ciphertext c, evaluates the blockcipher used by D.Enc on input key k and a value d_0, where the latter is any fixed element $d_0 \in \mathcal{D}$ in the cipher's domain, and assume K.Enc completely ignores the result. Even though this blockcipher evaluation is completely pointless and should not affect security of the overall design, for such a KEM our arguments would not work. Below, in the formal version of our theorem statement, we correspondingly restrict the set of considered KEMs to those that do not evaluate the blockcipher at all. This admittedly is a limitation of our result, but we believe it is a mild one. Indeed, all practical KEMs we are aware of do not (internally) invoke blockcipher operations at all. This holds in particular for Hashed ElGamal, PSEC-KEM, Cramer-Shoup KEM, and RSA-KEM. In the following theorem statement, if E is a blockcipher, we say a KEM is E-independent if no KEM algorithm evaluates E^+ or E^-.

We proceed with the statement and proof of our main theorem.

[4] We note that a typical proof of IND-CCA security of hybrid PKE requires the DEM to also offer some kind of confidentiality (e.g., OT-IND-CCA). A corresponding notion appears only implicitly in our theorem statement, as it follows from the DEM's simulatability (in the ideal cipher model).

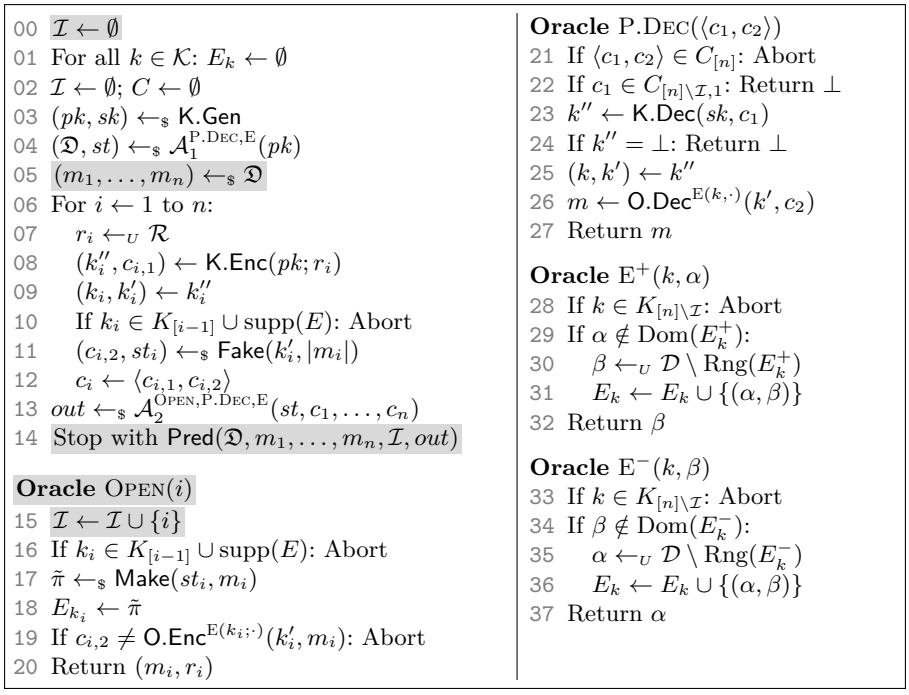

Fig. 17. Experiment in which adversary $(\mathcal{A}_1, \mathcal{A}_2)$ is run in, when embedding simulator $\mathcal{S}$ into the ideal game. Code in grey boxes is run by the ideal game. For $\mathcal{J} \subseteq [n]$ we denote $C_{\mathcal{J},1} := \{c_{j,1} \mid j \in \mathcal{J}\}$ and $K_{\mathcal{J}} := \{k_j \mid j \in \mathcal{J}\}$. Further, we denote $\text{supp}(E) := \{k \in \mathcal{K} \mid E_k \neq \emptyset\}$. This experiment corresponds to the last game G_6 in the sequence of games in the proof of Theorem 2.

Theorem 2. *Let* DEM *be a* $(\mathcal{K}, \mathcal{D})$*-permutation-driven DEM with corresponding oracle DEM* oDEM *and blockcipher* E*. Let* KEM *denote an* E*-independent KEM for the key space of the DEM. Let* PKE *denote the hybrid PKE scheme obtained when instantiating Construction 1 in Figure 3 with* KEM *and* DEM*.*

Let DEM *be* $(\tau, q_d, \epsilon_{ctxt})$*-OT-INT-CTXT secure and* KEM *be* $(\tau, q_d, \epsilon_{cca})$*-IND-CCA secure.*

If oDEM *is* ϵ_{sim}*-simulatable, then* PKE *is* $(\tau, \tau', q_d, q_{ic}, \epsilon)$*-SIM-SO-CCA secure where* ϵ *can be upper-bounded by*

$$\epsilon(n) \leq n \cdot \left(3 \cdot \epsilon_{cca} + \epsilon_{ctxt} + \epsilon_{sim} + 2 \cdot \frac{n + q_{ic} + q_d}{|\mathcal{K}|}\right)$$

and E *is modeled as an ideal cipher.*

See Section 1 for a proof sketch including the high-level ideas. We proceed with a detailed proof of Theorem 2.

Proof. For $\mathcal{J} \subseteq [n]$ let $C_{\mathcal{J},1}$ (resp. $K_{\mathcal{J}}$) denote the set $\{c_{j,1} \mid j \in \mathcal{J}\}$ (resp. $\{k_j \mid j \in \mathcal{J}\}$). We denote the set of keys $k \in \mathcal{K}$ where partial permutation E_k is not empty as $\text{supp}(E) := \{k \in \mathcal{K} \mid E_k \neq \emptyset\}$.

Fix any SIM-SO-CCA adversary $\mathcal{A}$ and consider the simulator $\mathcal{S}$ as given in Figure 17. Note that lines 01 – 04 correspond to the code of $\mathcal{S}_1$ from Figure 6, and lines 06 – 13 correspond to the code of $\mathcal{S}_2$. The code is enhanced by bookkeeping and abort events, while the explicit invocation of $\mathcal{S}_1$, $\mathcal{S}_2$ and their input/output behaviour is merged into the ideal game. Instructions in grey boxes are performed by the ideal game. Note that, technically, two a-priori distinct sets $\mathcal{I}$ are maintained: Simulator $\mathcal{S}_2$ keeps track of $\mathcal{A}$'s opening queries in a set $\mathcal{I}_{\mathcal{A}}$ while the ideal games tracks opening queries in another set $\mathcal{I}_{\mathcal{S}}$. As the simulator keeps both sets synchronised throughout its execution, we do not distinguish these set and write $\mathcal{I}$.

We show that $\mathcal{S}$, when run in the ideal game, can simulate the real game for $\mathcal{A}$. To this end we proceed in a sequence of experiments tracing $\mathcal{A}$'s advantage of distinguishing two consecutive games. The sequence interpolates between the real game (G_0 = r-SO-CCA, cf. Figure 5) and a simulated real game (cf. Figure 17) provided by the simulator when run in the ideal game (G_6 = i-SO-CCA, cf. Figure 4). The whole sequence of experiments is given in Figure 18. Lines ending with a range of games G_i – G_j (resp. G_i if $j = i$) are only executed when a game within the range is run.

Without loss of generality we assume that $\mathcal{A}$ does not make the same opening query twice. We proceed with detailed descriptions of experiments used in the proof of Theorem 2.

Experiments $\mathsf{G}_0 - \mathsf{G}_6$

00 For all $k \in \mathcal{K}$: $E_k \leftarrow \emptyset$
01 $\mathcal{I} \leftarrow \emptyset$; $C \leftarrow \emptyset$
02 bad $\leftarrow 0$ // G_4
03 $(pk, sk) \leftarrow_{\$} \mathsf{K.Gen}$
04 $(\mathfrak{D}, st) \leftarrow_{\$} \mathcal{A}_1^{\mathrm{P.Dec,E}}(pk)$
05 $(m_1, \ldots, m_n) \leftarrow_{\$} \mathfrak{D}$
06 For $i \leftarrow 1$ to n:
07 $\quad r_i \leftarrow_U \mathcal{R}$
08 $\quad (k_i'', c_{i,1}) \leftarrow \mathsf{K.Enc}(pk; r_i)$
09 $\quad (k_i, k_i') \leftarrow k_i''$
10 $\quad$ If $k_i \in K_{[i-1]} \cup \mathrm{supp}(E)$: Abort // $\mathsf{G}_2 - \mathsf{G}_6$
11 $\quad c_{i,2} \leftarrow \mathsf{O.Enc}^{\mathrm{E}(k_i;\cdot)}(k_i', m_i)$ // $\mathsf{G}_0 - \mathsf{G}_2$
12 $\quad (c_{i,2}, st_i) \leftarrow_{\$} \mathsf{Fake}(k_i', |m_i|)$ // $\mathsf{G}_3 - \mathsf{G}_6$
13 $\quad \tilde{\pi} \leftarrow_{\$} \mathsf{Make}(st_i, m_i)$ // $\mathsf{G}_3 - \mathsf{G}_5$
14 $\quad E_{k_i} \leftarrow \tilde{\pi}$ // $\mathsf{G}_3 - \mathsf{G}_5$
15 $\quad$ If $c_{i,2} \neq \mathsf{O.Enc}^{\mathrm{E}(k_i;\cdot)}(k_i', m_i)$: // $\mathsf{G}_3 - \mathsf{G}_5$
16 $\quad\quad$ Abort // $\mathsf{G}_3 - \mathsf{G}_5$
17 $\quad c_i \leftarrow \langle c_{i,1}, c_{i,2} \rangle$
18 $out \leftarrow_{\$} \mathcal{A}_2^{\mathrm{Open,P.Dec,E}}(st, c_1, \ldots, c_n)$
19 If bad: Abort // G_4
20 Stop with $\mathsf{Pred}(\mathfrak{D}, m_1, \ldots, m_n, \mathcal{I}, out)$

Oracle $\mathrm{Open}(i)$
21 $\mathcal{I} \leftarrow \mathcal{I} \cup \{i\}$
22 If $k_i \in K_{[i-1]} \cup \mathrm{supp}(E)$: Abort // G_6
23 $\tilde{\pi} \leftarrow_{\$} \mathsf{Make}(st_i, m_i)$ // G_6
24 $E_{k_i} \leftarrow \tilde{\pi}$ // G_6
25 If $c_{i,2} \neq \mathsf{O.Enc}^{\mathrm{E}(k_i;\cdot)}(k_i', m_i)$: Abort // G_6
26 Return (m_i, r_i)

Oracle $\mathrm{P.Dec}(\langle c_1, c_2 \rangle)$
27 If $\langle c_1, c_2 \rangle \in C_{[n]}$: Abort
28 If $c_1 \in C_{[n]\setminus\mathcal{I},1}$: // $\mathsf{G}_1 - \mathsf{G}_6$
29 $\quad$ Return $\perp$ // $\mathsf{G}_1 - \mathsf{G}_6$
30 $k'' \leftarrow \mathsf{K.Dec}(sk, c_1)$
31 If $k'' = \perp$: Return $\perp$
32 $(k, k') \leftarrow k''$
33 $m \leftarrow \mathsf{O.Dec}^{\mathrm{E}(k,\cdot)}(k', c_2)$
34 Return m

Oracle $\mathrm{E}^+(k, \alpha)$
35 If $k \in K_{[n]\setminus\mathcal{I}}$: // $\mathsf{G}_4 - \mathsf{G}_6$
36 $\quad$ bad $\leftarrow 1$ // G_4
37 $\quad$ Abort // $\mathsf{G}_5 - \mathsf{G}_6$
38 If $\alpha \notin \mathrm{Dom}(E_k^+)$:
39 $\quad \beta \leftarrow_U \mathcal{D} \setminus \mathrm{Rng}(E_k^+)$
40 $\quad E_k \leftarrow E_k \cup \{(\alpha, \beta)\}$
41 Return β

Oracle $\mathrm{E}^-(k, \beta)$
42 If $k \in K_{[n]\setminus\mathcal{I}}$: // $\mathsf{G}_4 - \mathsf{G}_6$
43 $\quad$ bad $\leftarrow 1$ // G_4
44 $\quad$ Abort // $\mathsf{G}_5 - \mathsf{G}_6$
45 If $\beta \notin \mathrm{Dom}(E_k^-)$:
46 $\quad \alpha \leftarrow_U \mathcal{D} \setminus \mathrm{Rng}(E_k^-)$
47 $\quad E_k \leftarrow E_k \cup \{(\alpha, \beta)\}$
48 Return α

Fig. 18. Experiments $\mathsf{G}_0 - \mathsf{G}_6$ used in the proof of Theorem 2. We write 'Abort' as an abbreviation for 'Stop with 0'.

Game G_0. The r-SO-CCA game as given in Figure 5.

Game G_1. Lines 28 and 29 are added: Any decryption query of the form $\langle c_1, c_2 \rangle$ is answered with $\perp$ if $c_1 \in C_{[n]\setminus\mathcal{I},1}$. That is, there exists $i \in [n]$ such that $c_1 = c_{i,1}$ and $\mathcal{A}$ did not query $\mathrm{Open}(i)$.

Claim. There exists an adversary $\mathcal{B}_{cca}$ that $(\tau, q_d, \epsilon_{cca})$-breaks the IND-CCA security of KEM and an adversary $\mathcal{B}_{ctxt}$ that $(\tau, q_d, \epsilon_{ctxt})$-breaks the OT-INT-CTXT security of DEM with $|\Pr[\mathsf{G}_0 \Rightarrow 1] - \Pr[\mathsf{G}_1 \Rightarrow 1]| \leq n \cdot (\epsilon_{cca} + \epsilon_{ctxt})$.

Proof. Games G_0 and G_1 proceed identically, until $\mathcal{A}$ submits a ciphertext $\langle c_1, c_2 \rangle$ to decryption where $c_1 \in C_{[n]\setminus\mathcal{I}}$ and $\mathsf{P.Dec}(sk, \langle c_1, c_2 \rangle) \neq \perp$. We fix some $i \in [n]$ and analyse the probability that $\mathcal{A}$ submits a ciphertext $\langle c_1, c_2 \rangle$ where $c_1 \in C_{\{i\}\setminus\mathcal{I}}$ and $\mathsf{P.Dec}(sk, \langle c_1, c_2 \rangle) \neq \perp$ we denote this event by '$\langle c_{i,1}, c_2 \rangle \nrightarrow \perp$'.

As a first step, we replace k_i'' as output by the i^{th} invocation of $\mathsf{K.Enc}$ with a uniformly random key. The price we pay is an additional summand of ϵ_{cca} in the bound on $\Pr[\langle c_{i,1}, c_2 \rangle \nrightarrow \perp]$ as shown by the following reduction run by adversary $\mathcal{B}_{cca}$: It uses its decapsulation oracle to answer decryption queries from $\mathcal{A}_1$. Receiving (c^*, k_b^*), $\mathcal{B}_{cca}$ parses $(k_b, k_b') \leftarrow k_b^*$ and computes all ciphertexts faithfully except for $c_i \leftarrow \langle c^*, \mathsf{O.Enc}^{\mathrm{E}(k_b;\cdot)}(k_b', m_i) \rangle$. Decryption queries $\langle c_1, c_2 \rangle$ by $\mathcal{A}_2$ are answered employing the decapsulation oracle for $c_1 \neq c^*$ and using key k_b^* otherwise.

The reduction perfectly simulates G_1 until $\mathcal{A}$ queries $\mathrm{Open}(i)$ which the reduction cannot answer. However, to bound the probability of event '$\langle c_{i,1}, c_2 \rangle \nrightarrow \perp$' happening, it suffices to make sure that the

reduction 'works' as long as the event can occur. Observe that '$\langle c_{i,1}, c_2\rangle \nrightarrow \perp$' cannot happen after query $\textsc{Open}(i)$.

We now show how to break the OT-INT-CTXT security of the DEM if '$\langle c_{i,1}, c_2\rangle \nrightarrow \perp$' happens. We construct $\mathcal{B}_{ctxt}$. The reduction performed by $\mathcal{B}_{ctxt}$ runs $\mathsf{K.Gen}$ and starts $\mathcal{A}_1(pk)$. Decryption queries are answered using sk. Once $\mathcal{A}_1$ outputs $\mathfrak{D}$, $\mathcal{B}_{ctxt}$ samples messages but submits m_i to the $\mathsf{D.Enc}$ oracle of its OT-INT-CTXT game to obtain a data encapsulation $c_2^* \leftarrow \mathsf{D.Enc}(k''_{\$}, m^*)$ under a random key $k''_{\$}$. Additionally, $\mathcal{B}_{ctxt}$ runs $\mathsf{K.Enc}$ to obtain (k, c_1^*) and sends $(c_1, \ldots, c_{i-1}, \langle c_1^*, c_2^*\rangle, \ldots, c_n)$ to $\mathcal{A}$. Adversary $\mathcal{B}_{ctxt}$ answers all further decryption queries on its own, unless the ciphertext is of the form $\langle c_1^*, c_2\rangle$ where it submits c_2 to the decapsulation oracle of the $\mathsf{OT\text{-}INT\text{-}CTXT}$ experiment. If it receives $\perp$, it returns $\perp$ to $\mathcal{A}_2$.

Clearly, $\mathcal{B}_{ctxt}$ wins the OT-INT-CTXT game when $\mathcal{A}$ submits a ciphertext that causes '$\langle c_{i,1}, c_2\rangle \nrightarrow \perp$' to happen.

We obtain $\Pr[\langle c_{i,1}, c_2\rangle \nrightarrow \perp] \leq \epsilon_{cca} + \epsilon_{ctxt}$. The claim follows from the union-bound over all $i \in [n]$. □

The next game hop ensures that (if it is not aborted) the i^{th} invocation of the oracle data encapsulation, i.e., $\mathsf{O.Enc}^{E(k_i;\cdot)}$, has access to an empty partial permutation E_{k_i}. This is a preparational step to ensure that later, when $\mathsf{O.Enc}$ is replaced with Fake and Make, the partial permutation output by Make can be embedded into E_{k_i}.

Game G_2. Line 10 is added. That is, G_2 aborts if the i^{th} iteration of $\mathsf{O.Enc}$ would have oracle access to a non-empty permutation $E(k_i;\cdot)$. [5]

Claim. There exists an adversary $\mathcal{B}_{cca}$ that $(\tau, q_d, \epsilon_{cca})$-breaks the IND-CCA security of KEM with $|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_2 \Rightarrow 1]| \leq n \cdot (\epsilon_{cca} + (n + q_{ic} + q_d) \,/\, |\mathcal{K}|)$.

Proof. We bound $\Pr[k_i \in K_{[i-1]} \cup \mathrm{supp}(E)]$ for fixed $i \in [n]$. Again, we use KEM's IND-CCA security to replace k_i'' output by the i^{th} invocation of $\mathsf{K.Enc}$ with a uniform key. We construct adversary $\mathcal{B}_{cca}$. It receives pk and starts $\mathcal{A}_1(pk)$. Decryption queries are answered using the decapsulation oracle. When $\mathcal{A}_1$ halts, $\mathcal{B}_{cca}$ requests its IND-CCA challenge (c^*, k_b^*) — let $(k_b, k_b') \leftarrow k_b^*$ — and runs the For loop 07. In the i^{th} iteration $\mathcal{B}_{cca}$ halts and returns 1 iff $k_b \in K_{[i-1]} \cup \mathrm{supp}(E)$. Clearly, the reduction is perfect until $\mathcal{B}_{cca}$ halts and we have $|\Pr[k_i \in K_{[i-1]} \cup \mathrm{supp}(E)] - \Pr[k_{\$} \in K_{[i-1]} \cup \mathrm{supp}(E)]| \leq \epsilon_{cca}$ where $k_{\$} \leftarrow_{\$} \mathcal{K}$.

Note that each decryption query or query to the ideal cipher oracles adds at most one element to $\mathrm{supp}(E)$, hence $|K_{[i-1]} \cup \mathrm{supp}(E)| \leq n + q_{ic} + q_d$. Thus, we obtain $\Pr[k_{\$} \in K_{[i-1]} \cup \mathrm{supp}(E)] \leq (n + q_{ic} + q_d) \,/\, |\mathcal{K}|$ and $\Pr[k_i \in K_{[i-1]} \cup \mathrm{supp}(E)] \leq \epsilon_{cca} + (n + q_{ic} + q_d) \,/\, (|\mathcal{K}|)$. The claim follows from the union-bound over $i \in [n]$. □

Game G_3. The faithful data encapsulation is replaced by algorithms Fake and Make. More precisely, for each iteration of the For loop (line 06) we replace the invocation $\mathsf{O.Dec}^{E(k_i;\cdot)}(k_i', m_i)$ (line 11) with running $\mathsf{Fake}(k_i', |m_i|)$ and $\mathsf{Make}(m_i)$ back to back (lines 12,13). E_{k_i} gets assigned partial permutation $\tilde{\pi}$ as output by Make (cf. line 14) and a check is performed whether E_{k_i} has been programmed 'consistently'; if not, experiment G_3 aborts (lines 15, 16).

Claim. $|\Pr[\mathsf{G}_2 \Rightarrow 1] - \Pr[\mathsf{G}_3 \Rightarrow 1]| \leq n \cdot \epsilon_{sim}$.

Proof. Fix $i \in [n]$. Due to the modifications in games G_1 and G_2 partial permutation E_{k_i} is empty at the time of invoking $\mathsf{O.Enc}$. Hence, once we replace $\mathsf{O.Enc}$ by Fake and Make, the partial permutation as output by Make can always be embedded into E_{k_i}. Particularly, partial permutations E_{k_i} accessed by $\mathsf{O.Enc}$ and $\tilde{\pi}$ output by Make are identically distributed when randomly extended to a full permutation on $\mathcal{D}$. We conclude that the abort in line 16 happens with probability at most ϵ_{sim} as oDEM is ϵ_{sim}-simulatable. The claim follows from the union-bound over all $i \in [n]$. □

Recall from the proof outline that, eventually, Make shall be run as part of the $\textsc{Open}$ procedure. The upcoming modifications ensure that partial permutation E_{k_i} remains empty until $\textsc{Open}(i)$ is queried.

[5] As of now, in the i^{th} iteration of the For loop, we have $K_{[i-1]} \subseteq \mathrm{supp}(E)$ as the invocation of $\mathsf{O.Enc}^{E(k_i;\cdot)}$ adds elements to E_{k_i}. Later, in game G_6, we do not invoke code that (implicitly) adds elements to E_{k_i} and rely on set $K_{[i-1]}$ to detect collisions amongst the (blockcipher) keys.

Game G_4. Line 02 is added to initialise a flag 'bad' as 0. Lines (35, 36) are added to the E^+ oracle, lines (42, 43) are added to the E^- oracle and line 19 is added. That is, if E^+ or E^- is queried on (k_i, z) for any z and $i \notin \mathcal{I}$, 'bad' is set to 1 and the game aborts after the execution of $\mathcal{A}_2$ (in line 19).

Claim. There exists an adversary $\mathcal{B}_{cca}$ that $(\tau, q_d, \epsilon_{cca})$-breaks the IND-CCA security of KEM with $|\Pr[\mathsf{G}_3 \Rightarrow 1] - \Pr[\mathsf{G}_4 \Rightarrow 1]| \leq n \cdot (\epsilon_{cca} + (q_{ic} + q_d)/|\mathcal{K}|)$.

Proof. Fix $i \in [n]$ and let '$k \in K_{\{i\}\setminus\mathcal{I}}$' denote the event that E^+ or E^- is queried on (k, z) where $k \in K_{\{i\}\setminus\mathcal{I}}$. (That is, the condition in lines 35 or 42 holds, even for $K_{\{i\}\setminus\mathcal{I}}$). Again, we replace key k_i'' output in the i^{th} invocation of $\mathsf{K.Enc}$ with a uniform key $(k_\$, k'_\$) \leftarrow k''_\$$. The reduction run by $\mathcal{B}_{cca}$ proceeds as in the proof to bridge G_0 and G_1. Here, $\mathcal{B}_{cca}$ halts after $\mathcal{A}_2$'s execution and outputs 1 iff bad $= 1$. Clearly $|\Pr[k \in K_{\{i\}\setminus\mathcal{I}}] - \Pr[k \in \{k_\$\} \setminus \mathcal{I}]| \leq \epsilon_{cca}$ for uniform $k_\$ \leftarrow_\$ \mathcal{K}$.

The reduction is perfect unless $\mathcal{A}_2$ queries $\textsc{Open}(i)$ which cannot be answered. Note that after query $\textsc{Open}(i)$, 'bad' cannot be set to 1 as $K_{\{i\}\setminus\mathcal{I}} = \emptyset$. Similarly to before, it suffices to guarantee the correctness of the simulation as long as the abort in line 19 can potentially happen.

Note that $k_\$$ is uniform from $\mathcal{A}$'s view: Only ciphertext $\langle c_{i,1}, c_{i,2}\rangle$ might contain information on $k_\$$ but $c_{i,1}$ is independent of $k_\$$ as it is sampled after $\mathsf{K.Enc}$ output $c_{i,1}$ and data encapsulation $c_{i,2}$ is independent of $k_\$$ as we run $\mathsf{Fake}(k'_i, m_i)$ to compute $c_{i,2}$. Thus, $\Pr[k \in \{k_\$\} \setminus \mathcal{I}] \leq (q_{ic} + q_d)/|\mathcal{K}|$ and collecting the probabilities and applying the union-bound gives the desired bound. □

Game G_5. Lines 37 and 44 are added. Instead of aborting after the execution of $\mathcal{A}_2$ if bad $= 1$, game G_5 aborts as soon as bad (as introduced in game G_4) is set to 1. Now obsolete lines 02, 19, 36 and 43 are removed for clarity.

Claim. $\Pr[\mathsf{G}_4 \Rightarrow 1] = \Pr[\mathsf{G}_5 \Rightarrow 1]$.

Proof. The claim follows from observing that game G_5 aborts in lines 37 or 44 if and only if game G_4 aborts in line 19. □

Game G_6. An abort event is added in line 22. The invocation of Make, the embedding of a partial permutation and the consistency check are moved from the For loop in lines 13 – 16 to the $\textsc{Open}$ oracle (lines 23 – 24).

Claim. $\Pr[\mathsf{G}_5 \Rightarrow 1] = \Pr[\mathsf{G}_6 \Rightarrow 1]$.

Proof. The abort event in line 22 is solely added for clarity but never met: Assume that line 22 would cause an abort, then the condition in line 10, or lines 35/42 would have been satisfied earlier. Hence, for all $i \in [n]$: a) in game G_5 partial permutation $E_{k_i} \leftarrow \tilde{\pi}$ as output by Make in line 13 is information-theoretically hidden from $\mathcal{A}$ until it queries $\textsc{Open}$ and b) in game G_6 partial permutation E_{k_i} remains empty until $\mathcal{A}$ queries $\textsc{Open}$. Thus, embedding partial permutation $\tilde{\pi}$ into E_{k_i} always succeeds. Further, moving the invocation of Make, the embedding and checking to the $\textsc{Open}$ oracle is completely oblivious to $\mathcal{A}$. □

We observe that the code as given in game G_6 in Figure 18 matches the code of the simulator as given in Figure 17.

The claim of Theorem 2 follows by collecting the probabilities. □

6 Conclusion

The most promising practical approach to public key encryption is through the hybrid KEM/DEM paradigm. Suitable KEMs include Hashed ElGamal, PSEC-KEM, Cramer-Shoup KEM, and RSA-KEM, and candidates for the DEM part are readily derived from the highly efficient encryption modes CTR, CBC, CCM, GCM standardized by NIST (to reach CCA security, the former two should be enhanced with a MAC, e.g., CMAC or HMAC). To compress the contribution of this paper into a single line: We effectively show that if any of these KEMs is combined with any of these DEMs in the sense of hybrid encryption, then the obtained PKE scheme offers a strong notion of selective opening security. Our result holds in the (heuristic) ideal cipher model for the underlying blockcipher. We thus recommend using modern blockciphers like AES as they come closest to meeting such requirements.

Acknowledgements

We thank the reviewers for their helpful feedback. Felix Heuer was funded by the German Research Foundation (DFG) as part of the priority program 1736 Big Data: Scalable Cryptography. Bertram Poettering was supported by ERC Project ERCC (FP7/615074).

References

1. M. Bellare, R. Dowsley, B. Waters, and S. Yilek. Standard security does not imply security against selective-opening. Cryptology ePrint Archive, Report 2011/581, 2011. http://eprint.iacr.org/2011/581.
2. M. Bellare, R. Dowsley, B. Waters, and S. Yilek. Standard security does not imply security against selective-opening. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 645–662. Springer, Heidelberg, Apr. 2012.
3. M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In A. Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, Heidelberg, Apr. 2009.
4. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Heidelberg, Dec. 2000.
5. M. Bellare and S. Yilek. Encryption schemes secure under selective opening attack. Cryptology ePrint Archive, Report 2009/101, 2009. http://eprint.iacr.org/2009/101.
6. F. Böhl, D. Hofheinz, and D. Kraschewski. On definitions of selective opening security. In M. Fischlin, J. Buchmann, and M. Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 522–539. Springer, Heidelberg, May 2012.
7. R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In B. S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 90–104. Springer, Heidelberg, Aug. 1997.
8. R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996.
9. J.-S. Coron, J. Patarin, and Y. Seurin. The random oracle model and the ideal cipher model are equivalent. In D. Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 1–20. Springer, Heidelberg, Aug. 2008.
10. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
11. D. Dachman-Soled. On minimal assumptions for sender-deniable public key encryption. In H. Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 574–591. Springer, Heidelberg, Mar. 2014.
12. C. Dwork, M. Naor, O. Reingold, and L. J. Stockmeyer. Magic functions. In *40th FOCS*, pages 523–534. IEEE Computer Society Press, Oct. 1999.
13. M. J. Dworkin. SP 800-38A: Recommendation for block cipher modes of operation: Methods and techniques. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2001. http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.
14. M. J. Dworkin. SP 800-38C: Recommendation for block cipher modes of operation: The CCM mode for authentication and confidentiality. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2007. http://csrc.nist.gov/publications/nistpubs/800-38C/SP800-38C_updated-July20_2007.pdf.
15. M. J. Dworkin. SP 800-38D: Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2007. http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf.
16. M. J. Dworkin. Addendum to SP 800-38A: Recommendation for block cipher modes of operation: Three variants of ciphertext stealing for CBC mode. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2010. http://csrc.nist.gov/publications/nistpubs/800-38a/addendum-to-nist_sp800-38A.pdf.
17. S. Even and Y. Mansour. A construction of a cipher from a single pseudorandom permutation. In H. Imai, R. L. Rivest, and T. Matsumoto, editors, *ASIACRYPT'91*, volume 739 of *LNCS*, pages 210–224. Springer, Heidelberg, Nov. 1993.
18. S. Fehr, D. Hofheinz, E. Kiltz, and H. Wee. Encryption schemes secure against chosen-ciphertext selective opening attacks. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 381–402. Springer, Heidelberg, May 2010.
19. G. Fuchsbauer, F. Heuer, E. Kiltz, and K. Pietrzak. Standard security does imply security against selective opening for markov distributions. In E. Kushilevitz and T. Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 282–305. Springer, Heidelberg, Jan. 2016.

20. C. Hazay, A. Patra, and B. Warinschi. Selective opening security for receivers. In T. Iwata and J. H. Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 443–469. Springer, Heidelberg, Nov. / Dec. 2015.
21. B. Hemenway, R. Ostrovsky, and A. Rosen. Non-committing encryption from ϕ-hiding. In Y. Dodis and J. B. Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 591–608. Springer, Heidelberg, Mar. 2015.
22. F. Heuer, T. Jager, E. Kiltz, and S. Schäge. On the selective opening security of practical public-key encryption schemes. In J. Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 27–51. Springer, Heidelberg, Mar. / Apr. 2015.
23. D. Hofheinz, T. Jager, and A. Rupp. Public-key encryption with simulation-based selective-opening security and compact ciphertexts. *IACR Cryptology ePrint Archive*, 2016:180, 2016.
24. D. Hofheinz, V. Rao, and D. Wichs. Standard security does not imply indistinguishability under selective opening. Cryptology ePrint Archive, Report 2015/792, 2015. http://eprint.iacr.org/2015/792.
25. D. Hofheinz and A. Rupp. Standard versus selective opening security: Separation and equivalence results. In Y. Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 591–615. Springer, Heidelberg, Feb. 2014.
26. J. Kilian and P. Rogaway. How to protect DES against exhaustive key search (an analysis of DESX). *Journal of Cryptology*, 14(1):17–35, 2001.
27. S. Liu and K. G. Paterson. Simulation-based selective opening CCA security for PKE from key encapsulation mechanisms. In J. Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 3–26. Springer, Heidelberg, Mar. / Apr. 2015.
28. C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In R. E. Ladner and C. Dwork, editors, *40th ACM STOC*, pages 187–196. ACM Press, May 2008.
29. P. Rogaway. Authenticated-encryption with associated-data. In V. Atluri, editor, *ACM CCS 02*, pages 98–107. ACM Press, Nov. 2002.
30. H. Wee. Zero knowledge in the random oracle model, revisited. In M. Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 417–434. Springer, Heidelberg, Dec. 2009.

A more thorough Definition of Simulatability

We give a more careful characterization of the properties (1) and (2) from Definition 11. While the original Fake and Make algorithms were randomised, here, for clarity, we switch to their deterministic counterparts with explicit randomness spaces.

Definition 12 (Simulatable oracle DEM). *For (deterministic) algorithms* Fake *and* Make *of the form*

$$\mathcal{K}' \times \mathbb{N} \times R_1 \to \mathsf{Fake} \to \mathcal{C} \times \Sigma \quad \textit{and} \quad \Sigma \times \mathcal{M} \times R_2 \to \mathsf{Make} \to \mathcal{PP}(\mathcal{D})$$

and $R = R_1 \times R_2$ *define composed algorithm*

$$\mathcal{K}' \times \mathcal{M} \times R \to \mathsf{FakeMake} \to \mathcal{PP}(\mathcal{D}) \times \mathcal{C}$$

and its two projections

$$\mathcal{K}' \times \mathcal{M} \times R \to \mathsf{FakeMake}_1 \to \mathcal{PP}(\mathcal{D}) \quad \textit{and} \quad \mathcal{K}' \times \mathcal{M} \times R \to \mathsf{FakeMake}_2 \to \mathcal{C}$$

such that for all $k' \in \mathcal{K}'$, $m \in \mathcal{M}$, $r_1 \in \mathcal{R}_1$, $r_2 \in \mathcal{R}_2$ *we have that*

$$\mathsf{Fake}(k', |m|, r_1) = (c, st) \wedge \mathsf{Make}(st, m, r_2) = \tilde{\pi}$$

implies that $\mathsf{FakeMake}(k', m, r_1 r_2) = (\tilde{\pi}, c)$ *and that*

$$\mathsf{FakeMake}_1(k', m, r_1 r_2) = \tilde{\pi} \quad \textit{and} \quad \mathsf{FakeMake}_2(k', m, r_1 r_2) = c \ .$$

For a partial permutation $\tilde{\pi} \in \mathcal{PP}(\mathcal{D})$ *let* $\tilde{\Pi} := \{\pi \in \mathcal{P}(\mathcal{D}) \mid \tilde{\pi} \subseteq \pi\}$ *denote the set of all extensions of* $\tilde{\pi}$ *to permutations* π *on* $\mathcal{D}$.

We say that an oracle DEM is ϵ-simulatable *by* Fake *and* Make *if for all all* $k' \in \mathcal{K}'$ *and* $m \in \mathcal{M}$ *we have that:*

(1) *Letting* $\tilde{\pi} \leftarrow_{\$} \mathsf{FakeMake}_1(k', m)$ *and uniformly picking a permutation from* $\tilde{\Pi}$ *yields a uniformly distributed element of* $\mathcal{P}(\mathcal{D})$.

(2) *It holds that*

$$\Pr[(\tilde{\pi}, c) \leftarrow_{\$} \mathsf{FakeMake}(k', m); \pi \leftarrow_{U} \tilde{\Pi}; c \neq \mathsf{O.Enc}^{\pi}(k', m)] \leq \epsilon \ ,$$

where the probability is taken over the choice of π *and the randomness of* FakeMake.

(3) *As in Definition 11.*