On the Selective Opening Security of Practical Public-Key Encryption Schemes

Felix Heuer, Tibor Jager, Eike Kiltz, and Sven Schäge

Horst Görtz Institute for IT-Security, Ruhr University Bochum, Germany
{felix.heuer,tibor.jager,eike.kiltz,sven.schaege}@rub.de

Abstract. We show that two well-known and widely employed public-key encryption schemes – RSA Optimal Asymmetric Encryption Padding (RSA-OAEP) and Diffie-Hellman Integrated Encryption Standard (DHIES), the latter one instantiated with a one-time pad, – are secure under (the strong, simulation-based security notion of) selective opening security against chosen-ciphertext attacks in the random oracle model. Both schemes are obtained via known generic transformations that transform relatively weak primitives (with security in the sense of one-wayness) to IND-CCA secure encryption schemes. We prove that selective opening security comes for free in these two transformations. Both DHIES and RSA-OAEP are important building blocks in several standards for public key encryption and key exchange protocols. They are the first practical cryptosystems that meet the strong notion of simulation-based selective opening (SIM-SO-CCA) security.

Keywords. public key encryption, selective opening security, OAEP, DHIES, SIM-SO-CCA

1 Introduction

Consider a set of clients $A_1, \ldots, A_n$ connecting to a server S. To encrypt a message m_i, each client A_i draws fresh randomness r_i and transmits ciphertext $c_i = \mathsf{Enc}_{pk_S}(m_i; r_i)$ to S. Assume an adversary observes these ciphertexts, and is then able to "corrupt" a subset of clients $\{A_i\}_{i\in\mathcal{I}}$, $\mathcal{I} \subseteq \{1, \ldots, n\}$, for instance by installing a malware on their computers. Then, for all $i \in \mathcal{I}$, the adversary learns not only the message m_i, but also the randomness r_i that A_i has used to encrypt m_i. Attacks of this type are called *selective-opening* (SO) attacks (under sender corruptions) and a central question in cryptography is whether the unopened ciphertexts remain secure.

At a first glance, one may be tempted to believe that security of the non-corrupted ciphertexts follows immediately, if the encryption scheme meets some standard security notion, like indistinguishability under chosen-plaintext (IND-CPA) or chosen-ciphertext (IND-CCA) attacks, due to the fact that each user A_i samples the randomness r_i independently from the other users. However, it has been observed [4, 15, 14, 3, 16] that this is not true in general, see e.g. [26] for an overview.

RESULTS ON SO SECURITY. Defining the right notion of security against selective opening attacks has proven highly non-trivial. There are three notions of security that are not polynomial-time equivalent to each other, two indistinguishability-based notions

usually denoted as weak IND-SO and (full) IND-SO security, and a simulation-based notion of selective opening security referred to as SIM-SO security. Previous results showed that SIM-SO-CCA and full IND-SO-CCA security are the strongest notions of security [11, 5, 26]. However, only SIM-SO-CCA has been realized so far [20, 24, 25]. Unfortunately, the existing constructions are very inefficient and rather constitute theoretical contributions. Intuitively, SIM-SO security says that for every adversary in the above scenario there exists a simulator which can produce the same output as the adversary without ever seeing any ciphertext, randomness, or the public key. It is noteworthy that unlike weak IND-SO security, which requires message distributions that support "efficient conditional re-sampling" (cf. [6]), SIM-SO is independent of the concrete distribution of the messages.

1.1 Our Contributions

In this paper we show that two important public key encryption systems are secure under the strong notion of SIM-SO-CCA security. Previous results only established IND-CCA security of the resulting schemes. Most notably, our results cover the well-known DHIES scheme, instantiated with a one-time pad, and RSA-OAEP. Our results show that SIM-SO security essentially comes for free in the random oracle model. This yields the first practical public key encryption schemes that meet the strong notion of SIM-SO-CCA security.

FIRST CONSTRUCTION: DHIES. The first construction we consider is a generalization of the well-known "Diffie-Hellman integrated encryption scheme" (DHIES) [1]. (DHIES or "Hashed ElGamal Encryption" uses a MAC to make plain ElGamal encryption IND-CCA secure.) This generic idea behind DHIES was formalized by Steinfeld, Baek, and Zheng [44] who showed how to build an IND-CCA secure public key encryption system from a key encapsulation mechanism (KEM) that is one-way under plaintext checking attacks (OW-PCA). OW-PCA is a comparatively weak notion of security in which the adversary's main task is to decapsulate a given encapsulation of some symmetric key. In addition to the public key, the adversary has only access to an oracle which checks, given a KEM key and a ciphertext, whether the ciphertext indeed constitutes an encapsulation of the KEM key under the public key. This construction is IND-CCA secure in the random oracle model [44]. We show that it is furthermore SIM-SO-CCA secure in the random oracle model. We stress that our result generically holds for the entire construction and therefore for any concrete instantiation that employs the one-time pad as symmetric encryption. Most importantly, it covers the well-known DHIES scheme (when instantiated with a one-time pad) that is contained in several public-key encryption standards like IEEE P1363a, SECG, and ISO 18033-2. DHIES is the de-facto standard for elliptic-curve encryption.

SECOND CONSTRUCTION: OAEP. The second construction of public key encryption schemes that we consider is the well-known Optimal Asymmetric Encryption Padding (OAEP) transformation [8]. OAEP is a generic transformation for constructing public-key encryption schemes from trapdoor permutations that was proposed by Bellare and Rogaway. Since then, it has become an important ingredient in many security protocols

and security standards like TLS [19, 40], SSH [23], S/MIME [39, 27], EAP [17], and Kerberos [34, 38].

We show that OAEP is SIM-SO-CCA secure when instantiated with a *partial-domain trapdoor permutation* (cf. Section 4.1). Since it is known [22] that the RSA permutation is partial-domain one-way under the RSA assumption, this implies that RSA-OAEP is SIM-SO-CCA secure under the RSA assumption. In fact, our result holds not only for trapdoor permutations, but for *injective* trapdoor functions as well.

Since SIM-SO-CCA security implies IND-CCA security, our proof also provides an alternative to the IND-CCA security proof of [22]. Interestingly, despite that we are analyzing security in a stronger security model, our proof seems to be somewhat simpler than the proof of [22], giving a more direct insight into which properties of the OAEP construction and the underlying trapdoor permutation make OAEP secure. This might be due to the fact that our proof is organized as a *sequence of games* [9].

Complementing the work of [22, 12, 2], our result gives new evidence towards the belief that the OAEP construction is sound, and that OAEP-type encryption schemes can be used securely in various practical scenarios.

1.2 Related Work

The problem of selective-opening attacks is well-known, and has already been observed twenty years ago [4, 15, 14, 3, 16]. The problem of constructing encryption schemes that are provably secure against this class of adversaries without random oracles has only been solved recently by Bellare, Hofheinz, and Yilek [6]. In [6], the authors show that *lossy* encryption [36] implies security against selective openings under chosen-plaintext attacks (SO-CPA). This line of research is continued in [24] by Hemenway *et al.*, who show that re-randomizable encryption and statistically hiding two-round oblivious transfer imply lossy encryption. From a cryptographic point of view, the above works solve the problem of finding SO-CPA secure encryption schemes, as there are several constructions of efficient lossy or re-randomizable encryption schemes, e.g. [36, 6, 24]. When it comes to selective openings under chosen-ciphertext attacks, the situation is somewhat different. Hemenway *et al.* [24], Fehr *et al.* [20], Hofheinz [25], and Fujisaki [21] describe SIM-SO-CCA secure encryption schemes which are all too inefficient for practical applications. More recently, an identity-based encryption scheme with selective-opening security was proposed [10]. It is noteworthy, that the most efficient public key encryption systems proven to be weak IND-SO secure do not meet the stronger notion of SIM-SO security. Lately, SIM-SO-CCA security for IBE has been achieved [32].

STATE-OF-THE-ART OF THE PROVABLE SECURITY OF OAEP. The OAEP construction was proved IND-CCA secure if the underlying trapdoor permutation is partial-domain one-way [8, 41, 22]. Since the RSA trapdoor permutation is a partial-domain one-way function, this yields the IND-CCA security of RSA-OAEP as well. Fischlin and Boldyreva [12] studied the security of OAEP when only one of the two hash functions is modelled as a random oracle, and furthermore showed that OAEP is non-malleable under chosen plaintext attacks for random messages without random oracles. The latter result was strengthened by Kiltz *et al.* [30], who proved the IND-CPA security of OAEP

without random oracles, when the underlying trapdoor permutation is *lossy* [36]. Since lossy encryption implies IND-SO-CPA security [6], this immediately shows that OAEP is IND-SO-CPA secure in the standard model. However, we stress that prior to our work it was not clear if OAEP meets the stronger notion of SIM-SO security, neither in the standard model nor in the random oracle. Backes *et al.* [2] showed that OAEP is secure under so-called *key-dependent message* attacks in the random oracle model.

There also exist a number of negative results [13, 31] showing the impossibility of instantiating OAEP without random oracles.

STATE-OF-THE-ART OF THE PROVABLE SECURITY OF DHIES. The IND-CCA security of DHIES in the random oracle model has been shown equivalent to the Strong Diffie-Hellman (sDH) assumption [1, 44].

2 Preliminaries

For $n \in \mathbb{N}$ let $[n] := \{1, \ldots, n\}$. For two strings μ, ν, we denote with $\mu||\nu$ the string obtained by concatenating μ with ν. If L is a set, then $|L|$ denotes the cardinality of L. We assume implicitly that any algorithm described in the sequel receives the unary representation 1^κ of the security parameter as input as its first argument. We say that an algorithm is a PPT algorithm, if it runs in probabilistic polynomial time (in κ). For a set A we denote the sampling of a uniform random element a by $a \xleftarrow{\$} A$, while we denote the sampling according to some distribution $\mathfrak{D}$ by $a \leftarrow \mathfrak{D}$.

2.1 Games

We present definitions of security and encryption schemes in terms of games and make use of sequences of games to proof our results. A game G is a collection of procedures/oracles $\{\text{INITIALIZE}, \mathrm{P}_1, \mathrm{P}_2, \ldots, \mathrm{P}_t, \text{FINALIZE}\}$ for $t \geq 0$, where P_1 to P_t and FINALIZE might require some input parameters, while INITIALIZE is run on the security parameter 1^κ. We implicitly assume that boolean flags are initialized to false, numerical types are initialized to 0, sets are initialized to $\emptyset$, while strings are initialized to the empty string ϵ. An adversary $\mathcal{A}$ is *run in game* G *(by challenger $\mathcal{C}$)*, if $\mathcal{A}$ calls INITIALIZE. During the game $\mathcal{A}$ may run the procedures P_i as often as allowed by the game. If a procedure P was called by $\mathcal{A}$, the output of P is returned to $\mathcal{A}$, except for the FINALIZE procedure. On $\mathcal{A}$'s call of FINALIZE the game ends and outputs whatever FINALIZE returns. The output *out* of a game G that runs $\mathcal{A}$ is denoted as $\mathsf{G}^{\mathcal{A}} \Rightarrow \mathit{out}$. If a game's output is either 0 or 1, $\mathcal{A}$ *wins* G if $\mathsf{G}^{\mathcal{A}} \Rightarrow 1$. Further, the *advantage* $\mathbf{Adv}(\mathsf{G}^{\mathcal{A}}, \mathsf{H}^{\mathcal{A}})$ of $\mathcal{A}$ in *distinguishing* games G and H is defined as $\big|\Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{H}^{\mathcal{A}} \Rightarrow 1]\big|$. For $\mathcal{A}$ run in G and $\mathcal{S}$ run in game H the *advantage* of $\mathcal{A}$ is defined as $|\Pr[\mathsf{G}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{H}^{\mathcal{S}} \Rightarrow 1]|$. Setting a boolean flag "ABORT..." to *true* implicitly aborts the adversary.

2.2 Public Key Encryption Schemes

Let $\mathfrak{M}$, $\mathfrak{R}$, $\mathfrak{C}$ be sets. We say that $\mathfrak{M}$ is the *message space*, $\mathfrak{R}$ is the *randomness space*, and $\mathfrak{C}$ is the *cipertext space*. A public key encryption scheme PKE =(PKEGen,Enc,Dec) consists of three polynomial-time algorithms.

- Gen generates, given the unary representation of the security parameter 1^κ, a key pair $(sk, pk) \leftarrow \mathsf{Gen}(1^\kappa)$, where pk defines $\mathfrak{M}$, $\mathfrak{R}$, and $\mathfrak{C}$.
- Given pk, and a message $m \in \mathfrak{M}$ Enc outputs an encryption $c \leftarrow \mathsf{Enc}_{pk}(m) \in \mathfrak{C}$ of m under the public key pk.
- The decryption algorithm Dec takes a secret key sk and a ciphertext $c \in \mathfrak{C}$ as input, and outputs a message $m = \mathsf{Dec}_{sk}(c) \in \mathfrak{M}$, or a special symbol $\perp \notin \mathfrak{M}$ indicating that c is not a valid ciphertext.

Notice, that Enc is a probabilistic algorithm; we make the used randomness only explicit when needed. In that case we write $c = \mathsf{Enc}(m; r)$ for $r \xleftarrow{\$} \mathfrak{R}$. We require the PKE to be correct, that is for all security parameters 1^κ, for all $(pk, sk) \leftarrow \mathsf{PKEGen}(1^\kappa)$, and for all $m \in \mathfrak{M}$ we have $\Pr[\mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m] = 1$

2.3 SIM-SO-CCA Security Definition

Definition 1 Let $\mathsf{PKE} := (\mathsf{PKEGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme, let $n = n(\kappa) > 0$ be a polynomially bounded function, $\mathfrak{D}$ a distribution over a message space, $\mathfrak{R}$ a randomness space and $\mathcal{R}$ a relation. We consider the following games, whereby an adversary $\mathcal{A}$ is run in the $\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA}_{\mathsf{PKE}}$ game (Figure 1), while a simulator $\mathcal{S} := \mathcal{S}(\mathcal{A})$ is run in the $\mathsf{IDEAL\text{-}SIM\text{-}SO\text{-}CCA}_{\mathsf{PKE}}$ game (Figure 2) . We

Procedure INITIALIZE	**Procedure** ENC($\mathfrak{D}$)	**Procedure** DEC(c)
$(pk, sk) \xleftarrow{\$} \mathsf{PKEGen}(1^\kappa)$	$(m_i)_{i\in[n]} \leftarrow \mathfrak{D}$	Return $\mathsf{Dec}_{sk}(c)$
Return pk	$(r_i)_{i\in[n]} \xleftarrow{\$} \mathfrak{R}$	**Procedure** OPEN(i)
Procedure FINALIZE(out)	$(c_i)_{i\in[n]} := \mathsf{Enc}_{pk}(m_i; r_i)$	$\mathcal{I} := \mathcal{I} \cup \{i\}$
Return $\mathcal{R}((m_i)_{i\in[n]}, \mathfrak{D}, \mathcal{I}, out)$	Return $(c_i)_{i\in[n]}$	Return (m_i, r_i)

Fig. 1: $\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA}_{\mathsf{PKE}}$ game.

Procedure INITIALIZE	**Procedure** ENC($\mathfrak{D}$)	**Procedure** OPEN(i)
Return ϵ	$(m_i)_{i\in[n]} \leftarrow \mathfrak{D}$	$\mathcal{I} := \mathcal{I} \cup \{i\}$
Procedure FINALIZE(out)	Return ϵ	Return m_i
Return $\mathcal{R}((m_i)_{i\in[n]}, \mathfrak{D}, \mathcal{I}, out)$		

Fig. 2: $\mathsf{IDEAL\text{-}SIM\text{-}SO\text{-}CCA}_{\mathsf{PKE}}$ game.

demand that $\mathcal{A}$ and $\mathcal{S}$ call ENC exactly one time before calling OPEN or FINALIZE.

Further, $\mathcal{A}$ is not allowed to call $\textsc{Dec}$ on any c_i. To an adversary $\mathcal{A}$, a simulator $\mathcal{S}$, a relation $\mathcal{R}$ and n we associate the advantage function

$$\mathbf{Adv}^{\mathsf{SIM\text{-}SO\text{-}CCA}}_{\mathsf{PKE}}(\mathcal{A},\mathcal{S},\mathcal{R},n,\kappa) := \\ |\Pr[\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA}^{\mathcal{A}}_{\mathsf{PKE}} \Rightarrow 1] - \Pr[\mathsf{IDEAL\text{-}SIM\text{-}SO\text{-}CCA}^{\mathcal{S}}_{\mathsf{PKE}} \Rightarrow 1]|.$$

PKE is SIM-SO-CCA secure if for every PPT adversary $\mathcal{A}$ and every PPT relation $\mathcal{R}$ there exists a PPT simulator $\mathcal{S}$ such that $\mathbf{Adv}^{\mathsf{SIM\text{-}SO\text{-}CCA}}_{\mathsf{PKE}}(\mathcal{A},\mathcal{S},\mathcal{R},n,\kappa) \leq \mathsf{negl}(\kappa)$.

3 Transformation from any OW-PCA secure KEM

3.1 Key Encapsulation Mechanisms and Message Authentication Codes

Definition 2 Let $\mathfrak{K}$ a key space. $\mathfrak{R}$ a randomness space, and $\mathfrak{C}$ a ciphertext space.

A Key Encapsulation Mechanism (KEM) consists of three PPT algorithms $\mathsf{KEM} = (\mathsf{KEMGen}, \mathsf{Encap}, \mathsf{Decap})$ defined to have the following syntax.

- KEMGen generates a key pair (pk,sk) on input 1^κ: $(pk, sk) \leftarrow \mathsf{KEMGen}(1^\kappa)$, where pk specifies $\mathfrak{K}$, $\mathfrak{R}$ and $\mathfrak{C}$.
- Encap is given pk and outputs a key $k \in \mathfrak{K}$ and an *encapsulation* $c \in \mathfrak{C}$ of k: $(c, k) \leftarrow \mathsf{Encap}_{pk}$.
- Given sk, Decap decapsulates $c \in \mathfrak{C}$: $k \leftarrow \mathsf{Decap}_{sk}(c)$, where $k \in \mathfrak{K}$.

We require correctness: for all $\kappa \in \mathbb{N}$, for all (pk, sk) generated by $\mathsf{KEMGen}(1^\kappa)$, and for all (c, k) output by Encap_{pk} we have $\Pr[\mathsf{Decap}_{sk}(c) = k] = 1$. We make the randomness used in Encap only explicit when needed. Without loss of generality we assume Encap to sample $k \xleftarrow{\$} \mathfrak{K}$, and $\mathfrak{K}$, $\mathfrak{C}$ to be exponentially large in the security parameter: $|\mathfrak{K}| \geq 2^\kappa$, $|\mathfrak{C}| \geq 2^\kappa$.

A KEM has unique encapsulations if for every $\kappa \in \mathbb{N}$ and every (pk, sk) output by $\mathsf{KEMGen}(1^\kappa)$ it holds that $\mathsf{Decap}_{sk}(c) = \mathsf{Decap}_{sk}(c') \Rightarrow c = c'$ for all $c, c' \in \mathfrak{C}$.

We introduce a security notion for KEMs that appeared in [35], namely *one-way* security in the presence of a *plaintext-checking oracle* (OW-PCA) amounting an adversary to test if some c is a *valid* encapsulation of a key k. That is, on input (c, k) and given the sk the oracle returns $\textsc{Check}_{sk}(c, k) := (\mathsf{Decap}_{sk}(c) \stackrel{?}{=} k) \in \{0, 1\}$. Since an indistinguishability based security notion is out of reach, once $\mathcal{A}$ is granted access to $\textsc{Check}$, we make use of a weaker security notion, given in the following definition.

Definition 3 Let $\mathsf{KEM} = (\mathsf{KEMGen}, \mathsf{Encap}, \mathsf{Decap})$ be a Key Encapsulation Mechanism and $\mathcal{A}$ an adversary run in the $\mathsf{OW\text{-}PCA}_{\mathsf{KEM}}$ game stated in Figure 3. We restrict the adversary to call $\textsc{Challenge}$ exactly one time and define $\mathcal{A}$'s advantage in winning the $\mathsf{OW\text{-}PCA}_{\mathsf{KEM}}$ game as

$$\mathbf{Adv}^{\mathsf{OW\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{A},\kappa) := \Pr[\mathsf{OW\text{-}PCA}^{\mathcal{A}}_{\mathsf{KEM}} \Rightarrow 1].$$

A KEM is OW-PCA secure, if $\mathbf{Adv}^{\mathsf{OW\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{A},\kappa)$ is negligible for all PPT $\mathcal{A}$.

Procedure $\textsc{Initialize}(1^\kappa)$	**Procedure** $\textsc{Challenge}$	**Procedure** $\textsc{Check}(k, c)$
$(pk, sk) \xleftarrow{\$} \mathsf{KEMGen}(1^\kappa)$	$(k^*, c^*) \xleftarrow{\$} \mathsf{Encap}_{pk}$	Return $(\mathsf{Decap}(c) \stackrel{?}{=} k)$
Return pk	Return c^*	**Procedure** $\textsc{Finalize}(k)$
		Return $(k \stackrel{?}{=} k^*)$

Fig. 3: $\mathsf{OW\text{-}PCA}_{\mathsf{KEM}}$ game

Definition 4 Let $\mathfrak{M}$ be a message space and let $\mathcal{T}$ be a set (*tag space*). A Message Authentication Code MAC consists of the following three PPT algorithms $\mathsf{MAC} = (\mathsf{MACGen}, \mathsf{Tag}, \mathsf{Vrfy})$, whereby

- MACGen generates a key k on input 1^κ: $k \leftarrow \mathsf{MACGen}(1^\kappa)$.
- Tag_k computes a tag $t \in \mathcal{T}$ for a given message $m \in \mathfrak{M}$: $t \leftarrow \mathsf{Tag}_k(m)$.
- Given a message $m \in \mathfrak{M}$ and a tag $t \in \mathcal{T}$, Vrfy_k, outputs a bit: $\{0,1\} \leftarrow \mathsf{Vrfy}_k(m,t)$.

We require MAC to be correct: For all $\kappa \in \mathbb{N}$, all keys k generated by $\mathsf{MACGen}(1^\kappa)$, all $m \in \mathfrak{M}$ and all tags computed by $\mathsf{Tag}_k(m)$ we have $\Pr[\mathsf{Vrfy}_k(m, \mathsf{Tag}_k(m)) = 1] = 1$. For a fixed MAC and k, given message m we call a tag t / the tuple (m,t) valid, if $\mathsf{Vrfy}_k(m,t) = 1$.

Definition 5 For an adversary $\mathcal{A}$ and a MAC $\mathsf{MAC} := (\mathsf{MACGen}, \mathsf{Tag}, \mathsf{Vrfy})$ we consider the $\mathsf{sUF\text{-}OT\text{-}CMA}_{\mathsf{MAC}}$ (strongly unforgeable under one-time chosen message attacks) game, where $\mathcal{A}$ is allowed to call $\textsc{Tag}$ at most once.

Procedure $\textsc{Initialize}(1^\kappa)$	**Procedure** $\textsc{Tag}(m)$
$k \xleftarrow{\$} \mathsf{MACGen}(1^\kappa)$	$t \leftarrow \mathsf{Tag}_k(m)$
Return ϵ	Return t
Procedure $\textsc{Finalize}(m^*, t^*)$	**Procedure** $\textsc{Vrfy}(\tilde{m}, \tilde{t})$
Return $(\mathsf{Vrfy}_k(m^*, t^*) \wedge (m^*, t^*) \neq (m, t))$	Return $\mathsf{Vrfy}_k(\tilde{m}, \tilde{t})$

We define the advantage of $\mathcal{A}$ run in the $\mathsf{sUF\text{-}OT\text{-}CMA}_{\mathsf{MAC}}$ game as

$$\mathbf{Adv}^{\mathsf{sUF\text{-}OT\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{A}, \kappa) := \Pr[\mathsf{sUF\text{-}OT\text{-}CMA}^{\mathcal{A}}_{\mathsf{MAC}} \Rightarrow 1].$$

MAC is sUF-OT-CMA secure, if $\mathbf{Adv}^{\mathsf{sUF\text{-}OT\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{A}, \kappa) \leq \mathsf{negl}(\kappa)$ holds for all PPT adversaries $\mathcal{A}$.

Note that we only require one-time security, so a sUF-OT-CMA secure MAC can be constructed information-theoretically.

3.2 The Transformation

Before we prove our results on the selective-opening security of schemes built from KEMs, we recall a well known transformation ([44]) to turn a given KEM into a PKE scheme. Notice, that we instantiated the symmetric encryption with a one-time-pad.

Let $\mathsf{KEM} = (\mathsf{KEMGen}, \mathsf{Encap}, \mathsf{Decap})$ be a KEM, $\mathcal{H}$ a family of hash functions, and let $\mathsf{MAC} = (\mathsf{MACGen}, \mathsf{Tag}, \mathsf{Vrfy})$ be a MAC. The public-key encryption scheme $\mathsf{PKE}_{\mathsf{KEM},\mathsf{MAC}}$ obtained by the transformation is given in Figure 4.

Procedure $\mathrm{PKEGEN}(1^\kappa)$	**Procedure** $\mathrm{ENC}(m)$	**Procedure** $\mathrm{DEC}(c^{(1)}, c^{(2)}, c^{(3)})$
$(pk_{\mathsf{KEM}}, sk_{\mathsf{KEM}}) \xleftarrow{\$} \mathsf{KEMGen}(1^\kappa)$	$(k, c^{(1)}) \xleftarrow{\$} \mathsf{Encap}_{pk_{\mathsf{KEM}}}$	$k \leftarrow \mathsf{Decap}_{sk_{\mathsf{KEM}}}(c^{(1)})$
$H \xleftarrow{\$} \mathcal{H}$	$(k^{sym}, k^{mac}) := H(k)$	$(k^{sym}, k^{mac}) := H(k)$
$pk := (pk_{\mathsf{KEM}}, H)$	$c^{(2)} := k^{sym} \oplus m$	if $\mathsf{Vrfy}_{k^{mac}}(c^{(2)}, c^{(3)}) = 1$
$sk := sk_{\mathsf{KEM}}$	$c^{(3)} := \mathsf{Tag}_{k^{mac}}(c^{(2)})$	Return $c^{(2)} \oplus k^{sym}$
Return pk	Return $(c^{(1)}, c^{(2)}, c^{(3)})$	else
		Return $\perp$

Fig. 4: Transformation $\mathsf{PKE}_{\mathsf{KEM},\mathsf{MAC}}$ from KEM and MAC to PKE.

It is well known, that the given construction turns a OW-PCA KEM into a IND-CCA secure PKE scheme in the random oracle model [44]. Our next theorem strengthens this results by showing that $\mathsf{PKE}_{\mathsf{KEM},\mathsf{MAC}}$ is even SIM-SO-CCA secure.

Theorem 6 *Let* KEM *be a* OW-PCA *secure KEM with unique encapsulations and let* MAC *be a* sUF-OT-CMA *secure MAC. Then* $\mathsf{PKE}_{\mathsf{KEM},\mathsf{MAC}}$ *is* SIM-SO-CCA *secure in the random oracle model. In particular, for any adversary $\mathcal{A}$ run in the* REAL-SIM-SO-$\mathsf{CCA}_{\mathsf{PKE}_{\mathsf{KEM},\mathsf{MAC}}}$ *game, that issues at most $q_h \leq 2^{\kappa-1}$ hash and $q_d \leq 2^{\kappa-1}$ decryption queries and obtains n ciphertexts, and every* PPT *relation $\mathcal{R}$, there exists a simulator $\mathcal{S}$, a forger $\mathcal{F}$ run in the* sUF-OT-$\mathsf{CMA}_{\mathsf{MAC}}$ *game, and an adversary $\mathcal{B}$ run in the* OW-$\mathsf{PCA}_{\mathsf{KEM}}$ *game with roughly the same running time as $\mathcal{A}$ such that*

$$\mathbf{Adv}^{\mathsf{SIM\text{-}SO\text{-}CCA}}_{\mathsf{PKE}_{\mathsf{KEM},\mathsf{MAC}}}(\mathcal{A}, \mathcal{S}, \mathcal{R}, n, \kappa) \leq$$

$$n \cdot \left(\frac{q_h + q_d}{2^{\kappa-1}} + \mathbf{Adv}^{\mathsf{sUF\text{-}OT\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{F}, \kappa) + \mathbf{Adv}^{\mathsf{OW\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{B}, \kappa) \right). \quad (1)$$

Let us have a high-level look at our proof. Up to some small syntactical changes G_0 constitutes of the REAL-SIM-SO-$\mathsf{CCA}^{\mathcal{A}}_{\mathsf{PKE}_{\mathsf{KEM},\mathsf{MAC}}}$ game.

The later simulator $\mathcal{S}$ will provide $\mathcal{A}$ with message-independent dummy encryptions c_i. This allows $\mathcal{S}$ to claim that $c_i = \mathsf{Enc}_{k_i}(m_i, r_i)$ for some arbitrary m_i after sending c_i to $\mathcal{A}$, if $\mathcal{A}$ should decide to open c_i. Game G_2 introduces the dummy encryptions, while G_1 serves as a preparational step.

After calling $\mathrm{ENC}(\mathfrak{D})$, $\mathcal{A}$ is allowed to make OPEN, HASH and DEC queries in an arbitrary order. Assume, that $\mathcal{A}$ did not[1] query $\mathrm{OPEN}(i)$ before calling $\mathrm{HASH}(k_i)$ or issuing a valid decryption query $(c_i^{(1)}, \cdot, \cdot)$. Since the indexset of opened messages $\mathcal{I}$ is part of $\mathcal{A}$'s output $\mathcal{S}$ wants to simulate, $\mathcal{S}$ may not query $\mathrm{OPEN}(i)$ if $\mathcal{A}$ did not make the same call. Neither can $\mathcal{S}$ answer such a query, since it would fix k_i^{sym} and thereby

[1] Neither $\mathrm{HASH}(k_i)$ nor $\mathrm{DEC}(c_i^{(1)}, \cdot, \cdot)$ queries are a tripping hazard, once $\mathcal{A}$ called $\mathrm{OPEN}(i)$.

m_i before $\mathcal{A}$ made a potential $\textsc{Open}(i)$ query. Therefore, we need to block $\textsc{Hash}(k_i)$ and valid $\mathsf{Dec}(c_i^{(1)}, \cdot, \cdot)$ queries, if $\mathcal{A}$ did not call $\textsc{Open}(i)$ before. Considering valid decryption queries, these two cases can occur: 1) $H(k_i)$ is already defined, or 2) $H(k_i)$ not defined. Game G_3 takes care of case 2), while we block $\mathcal{A}$'s hash queries $\textsc{Hash}(k_i)$ for unopened c_i in game G_4 - that is, ruling out case 1) as well.

Proof. Let q_h be the number of hash queries and let q_d be the number of decryption queries issued by $\mathcal{A}$, let $n = n(\kappa)$ be a polynomial in κ. For $i \in [n]$ let: m_i denote the i^{th} message sampled by the challenger, r_i the i^{th} randomness used by Encap: $(k_i, c_i^{(1)}) \leftarrow \mathsf{Encap}(r_i)$, $(k_i^{sym}, k_i^{mac}) \leftarrow H(k_i)$ the i^{th} key-pair generated by hashing k_i and $c_i := (c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$ the i^{th} ciphertext. Without loss of generality, the games samples $(r_i)_{i\in[n]}$ as part of $\textsc{Initialize}$. We proceed with a sequence of games which is given in pseudocode in Figure 5.

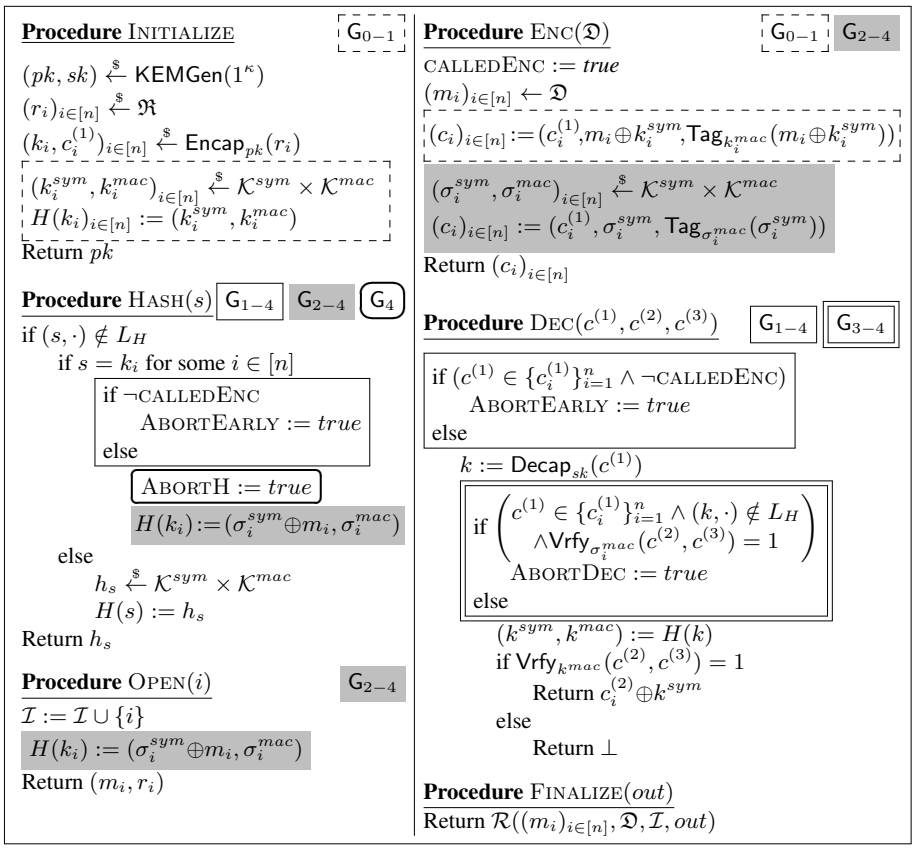

Fig. 5: Sequence of games G_0 to G_4. Boxed code is only executed in the games indicated by the game names given in the same box style at the top right of every procedure.

Game 0. We model H as a random oracle. Challenger $\mathcal{C}_0$ keeps track of issued calls (either by the game or $\mathcal{A}$) of $\textsc{Hash}(s)$ by maintaining a list L_H. For a query s, $\textsc{Hash}(s)$ returns h_s if there is an entry $(s, h_s) \in L_H$, otherwise $\textsc{Hash}$ samples h_s at random, adds (s, h_s) to L_H, and returns h_s; we write $H(s) := h_s$ only and implicitly assume an update operation $L_H := L_H \cup \{(s, h_s)\}$ to happen in the background.
We introduce small syntactical changes: Challenger $\mathcal{C}_0$ samples $(k_i^{sym}, k_i^{mac})_{i\in[n]}$ uniformly random and sets $(H(k_i))_{i\in[n]} := (k_i^{sym}, k_i^{mac})$ while $\textsc{Initialize}$ is run. Additionally, G_0 runs Encap_{pk} to generate $(k_i, c_i^{(1)})_{i\in[n]}$ during $\textsc{Initialize}$.
Claim 0. $\mathbf{Adv}(\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA}^{\mathcal{A}}_{\mathsf{PKE}_{\mathsf{KEM},\mathsf{MAC}}}, \mathsf{G}_0^{\mathcal{A}}) = 0$.

Proof. Apparently, it makes no difference if the challenger samples $(r_i)_{i\in[n]}$ and runs $\mathsf{Encap}(r_i)$ on demand as part of $\textsc{Enc}$ or in advance while $\textsc{Initialize}$ is run.
Since H is modeled as a random oracle, $H(s)$ is sampled uniformly random for every fresh query $\textsc{Hash}(s)$. Therefore $\mathcal{C}_0$ does not change the distribution by sampling (k_i^{sym}, k_i^{mac}) in the first place and setting $H(k_i) := (k_i^{sym}, k_i^{mac})$ afterwards.

Game 1. We add an abort condition. Challenger $\mathcal{C}_1$ raises the event $\textsc{AbortEarly}$ and aborts[2] $\mathcal{A}$, if $\mathcal{A}$ did not call $\textsc{Enc}$ before calling either $\textsc{Hash}(k_i)$ or $\textsc{Dec}(c_i^{(1)}, \cdot, \cdot)$ for some $i \in [n]$.
Claim 1. $\mathbf{Adv}(\mathsf{G}_0^{\mathcal{A}}, \mathsf{G}_1^{\mathcal{A}}) \leq n \cdot (q_h + q_d) \cdot 2^{-(\kappa-1)}$.

Proof. Since games G_0 and G_1 are identical until $\textsc{AbortEarly}$ is raised, it follows that $\mathbf{Adv}(\mathsf{G}_0^{\mathcal{A}}, \mathsf{G}_1^{\mathcal{A}}) \leq \Pr[\textsc{AbortEarly}]$. Let $\textsc{viaHash}$ and $\textsc{viaDec}$ be the events that $\textsc{AbortEarly}$ was caused by either a hash or a decryption query of $\mathcal{A}$. Let s_i denote the i^{th} hash and $d_i = (d_i^{(1)}, d_i^{(2)}, d_i^{(3)})$ the i^{th} decryption query of $\mathcal{A}$. It holds that

$$\Pr[\textsc{AbortEarly}] = \Pr[\textsc{viaHash}] + \Pr[\textsc{viaDec}]$$

$$\leq \Pr[s_1 \in \{k_i\}_{i=1}^n] + \sum_{i=2}^{q_h} \Pr[s_i \in \{k_i\}_{i=1}^n | \bigwedge_{j=1}^{i-1} s_j \notin \{k_i\}_{i=1}^n]$$

$$+ \Pr[d_i^{(1)} \in \{c_i^{(1)}\}_{i=1}^n] + \sum_{i=2}^{q_d} \Pr[d_i^{(1)} \in \{c_i^{(1)}\}_{i=1}^n | \bigwedge_{j=1}^{i-1} d_j^{(1)} \notin \{c_i^{(1)}\}_{i=1}^n]$$

$$= \sum_{i=1}^{q_h} \frac{n}{2^\kappa - (i-1)} + \sum_{i=1}^{q_d} \frac{n}{2^\kappa - (i-1)} \leq \sum_{i=1}^{q_h} \frac{n}{2^\kappa - q_h} + \sum_{i=1}^{q_d} \frac{n}{2^\kappa - q_d} \leq \frac{n(q_h + q_d)}{2^{\kappa-1}}.$$

Above holds since Encap samples $k \xleftarrow{\$} \mathfrak{K}$ and KEM has unique encapsulations.

Game 2. We change the encryption procedure and answer hash queries in a different way. $\mathcal{C}_2$ does not program $H(k_i)$ for $i \in [n]$ anymore. $\textsc{Enc}$ still samples m_i, and samples $\sigma_i^{sym} \xleftarrow{\$} \mathcal{K}^{sym}$, $\sigma_i^{mac} \xleftarrow{\$} \mathcal{K}^{mac}$, to compute $c_i = \mathsf{Enc}_{k_i}(m_i) := (c_i^{(1)}, \sigma_i^{sym},$

[2] Notice, that $\mathcal{C}_1$ aborts even if such a decryption query is invalid.

$\mathsf{Tag}_{\sigma_i^{mac}}(\sigma_i^{sym})$). If $\mathcal{A}$ should call $\textsc{Hash}(k_i)$ for $i \in [n]$ or $\textsc{Open}(i)$, the challenger programs $H(k_i) := (\sigma_i^{sym} \oplus m_i, \sigma_i^{mac})$. Keep in mind that as from now $(k_i, \cdot) \notin L_H$ implies that $\textsc{Open}(i)$ was not called.
Claim 2. $\mathbf{Adv}(\mathsf{G}_1^{\mathcal{A}}, \mathsf{G}_2^{\mathcal{A}}) = 0$.

Proof. Assuming that $\textsc{AbortEarly}$ does not happen in game G_2, the keys k_i^{sym} and k_i^{mac} are still uniformly random when $\mathcal{A}$ calls Enc. Therefore $(c_i^{(2)})_{i\in[n]} = m_i \oplus k_i^{sym}$ is uniform and $(c_i^{(3)})_{i\in[n]}$ is a valid tag of a uniformly random message under a key from the uniform distribution. Consequently, challenger $\mathcal{C}_2$ can sample $(c_i^{(2)})_{i\in[n]} := \sigma_i^{sym}$ uniformly and can compute the tags using a uniform key σ_i^{mac} without changing the distribution of the encryptions $(c_i)_{i\in[n]}$.
$\mathcal{C}_2$ does not program $H(k_i)$ for $i \in [n]$ anymore, but has to keep H consistent. If $\mathcal{A}$ calls $\textsc{Hash}(k_i)$ or $\textsc{Open}(i)$, $\mathcal{C}_2$ sets $H(k_i) := (\sigma_i^{sym} \oplus m_i, \sigma_i^{mac})$.

Game 3. We add another abort condition. If $\mathcal{A}$ already called $\textsc{Enc}$, issues a decryption query $(c_i^{(1)}, c^{(2)}, c^{(3)}) \notin \{c_i\}_{i=1}^n$, where $H(k_i)$ is not defined, and $\mathsf{Vrfy}_{\sigma_i^{mac}}(c_i^{(1)}, c^{(2)}, c^{(3)})$ verifies, challenger $\mathcal{C}_3$ raises $\textsc{AbortDec}$ and aborts $\mathcal{A}$.
Claim 3. $\mathbf{Adv}(\mathsf{G}_2^{\mathcal{A}}, \mathsf{G}_3^{\mathcal{A}}) \leq n \cdot \mathbf{Adv}^{\mathsf{sUF\text{-}OT\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{F}, \kappa)$.

Proof. Games G_2 and G_3 are identical until $\textsc{AbortDec}$ happens, it suffices to bound $\Pr[\textsc{AbortDec}]$.

Let $\mathsf{MAC} := (\mathsf{MACGen}, \mathsf{Tag}, \mathsf{Vrfy})$ be the MAC used by the $\mathsf{sUF\text{-}OT\text{-}CMA}$ challenger. We construct an adversary $\mathcal{F}$ against the $\mathsf{sUF\text{-}OT\text{-}CMA}$ security of MAC having success probability $\Pr[\textsc{AbortDec}]/n$. The reduction is straight forward: $\mathcal{F}$ runs adversary $\mathcal{A}$ as in game G_3, but picks $i^* \xleftarrow{\$} [n]$ during $\textsc{Initialize}$. Computing the i^{*th} ciphertext, $\mathcal{F}$ queries its $\mathsf{sUF\text{-}OT\text{-}CMA}$ challenger for $t^* := \textsc{Tag}(\sigma_{i^*}^{sym})$ instead of using its own $\textsc{Tag}$ procedure and sends $(c_i)_{i\in[n]}$ to $\mathcal{A}$. If $\mathcal{A}$ should call $\textsc{Open}(i^*)$, challenger C_3 apparently was unlucky in hiding its own challenge and aborts the adversary. Querying its $\textsc{Vrfy}_k(\cdot,\cdot)$ oracle, $\mathcal{F}$ can detect when $\mathcal{A}$ issues a valid query $\textsc{Dec}(c_i^{(1)}, c^{(2)}, c^{(3)})$ for some $i \in [n]$, returns $(c^{(2)}, c^{(3)})$ to its $\mathsf{sUF\text{-}OT\text{-}CMA}$ challenger and aborts $\mathcal{A}$.

Assume that $\textsc{AbortDec}$ happens, i.e. $\mathcal{A}$ makes a valid decryption query $(c_i^{(1)}, c^{(2)}, c^{(3)}) \notin \{c_i\}_{i\in[n]}$, while $H(k_i)$ is still undetermined. Notice, that we must not allow $H(k_{i^*})$ to be fixed since $k_{i^*}^{mac}$ is only known to the $\mathsf{sUF\text{-}OT\text{-}CMA}$ challenger. Let $(c_i^{(1)}, \tilde{c}^{(2)}, \tilde{c}^{(3)}) \in \{c_i\}_{i\in[n]}$ be the ciphertext c_i, whose first component matches the first entry of $\mathcal{A}$'s valid decryption query. Hence, $c^{(3)}$ is either a new valid tag for $\tilde{c}^{(2)}$ or $c^{(3)}$ is a valid tag for a "new" message $c^{(2)}$, since $(c^{(2)}, c^{(3)}) \neq (\tilde{c}^{(2)}, \tilde{c}^{(3)})$. In both cases $\mathcal{F}$ wins its $\mathsf{sUF\text{-}OT\text{-}CMA}$ challenge by returning $(c^{(2)}, c^{(3)})$, if $\mathcal{F}$ picks the right challenge ciphertext to embed t^*. The claim follows by rearranging

$$\mathbf{Adv}^{\mathsf{sUF\text{-}OT\text{-}CMA}}_{\mathsf{MAC}}(\mathcal{F}, \kappa) \geq \Pr[\textsc{AbortDec}]/n.$$

Game 4. We add one more abort condition. Challenger $\mathcal{C}_4$ raises the event $\textsc{AbortH}$ if $\mathcal{A}$ already called $\textsc{Enc}$, issues a hash query $\textsc{Hash}(k_i)$ for $i \in [n]$ and did not call $\textsc{Open}(i)$ before.
Claim 4. $\mathbf{Adv}(\mathsf{G}_3^{\mathcal{A}}, \mathsf{G}_4^{\mathcal{A}}) \leq n \cdot \mathbf{Adv}^{\mathsf{OW\text{-}PCA}}_{\mathsf{KEM}}(\mathcal{B}, \kappa)$.

Procedure INITIALIZE:
$pk \xleftarrow{\$} \text{INITIALIZE}_{\mathsf{OW\text{-}PCA}}$
$i^* \xleftarrow{\$} [n]$
$(r_i)_{i\in[n]\setminus\{i^*\}} \xleftarrow{\$} \mathfrak{R}$
$(k_i, c_i^{(1)})_{i\in[n]\setminus\{i^*\}} \leftarrow \mathsf{Encap}_{pk}(r_i)$
$c_{i^*}^{(1)} \xleftarrow{\$} \text{CHALLENGE}$
Return pk

Procedure HASH(s)
if $(s,\cdot) \notin L_H$
  if $s = k_i$ for some $i \in [n] \setminus \{i^*\}$
    if $\neg$CALLEDENC
      ABORTEARLY $:= true$
    else
      ABORTH $:= true$
  else
    if CHECK$(s, c_{i^*}^{(1)}) = 1$
      $\text{FINALIZE}_{\mathsf{OW\text{-}PCA}}(s)$
      ABORTH $:= true$
    else
      if $\left(\begin{array}{l}\exists (c^{(1)},k^{sym},k^{mac}) \in H_{patch} \\ \text{s.t. CHECK}(s, c^{(1)}) = 1)\end{array}\right)$
        $H(s) := (k^{sym}, k^{mac})$
      else
        $h_s \xleftarrow{\$} \mathcal{K}^{sym} \times \mathcal{K}^{mac}$
        $H(s) := h_s$
Return $H(s)$

Procedure OPEN(i)
if $i = i^*$
  ABORT $:= true$
else
  $\mathcal{I} := \mathcal{I} \cup \{i\}$
  $H(k_i) := (\sigma_i^{sym} \oplus m_i, \sigma_i^{mac})$
  Return (m_i, r_i)

Procedure ENC$(\mathfrak{D})$:
CALLEDENC $:=$ *true*
$(m_i)_{i\in[n]} \leftarrow \mathfrak{D}$
$(\sigma_i^{sym}, \sigma_i^{mac})_{i\in[n]} \xleftarrow{\$} \mathcal{K}^{sym} \times \mathcal{K}^{mac}$
$(c_i)_{i\in[n]} := (c_i^{(1)}, \sigma_i^{sym}, \mathsf{Tag}_{\sigma_i^{mac}}(\sigma_i^{sym}))$
Return $(c_i)_{i\in[n]}$

Procedure DEC$(c^{(1)}, c^{(2)}, c^{(3)})$
if $(c^{(1)} \in \{c_i^{(1)}\}_{i=1}^n \wedge \neg$CALLEDENC$)$
  ABORTEARLY $:= true$
else
  if $c^{(1)} \in \{c_i^{(1)}\}_{i=1}^n$
    if $\mathsf{Vrfy}_{\sigma_i^{mac}}(c^{(2)}, c^{(3)}) = 1$
      ABORTDEC $:= true$
    else
      Return $\perp$
  else
    if $\left(\begin{array}{l}\exists\, (s,\cdot,\cdot) \in L_H \text{ s.t.} \\ \text{CHECK}(s, c^{(1)}) = 1\end{array}\right)$
      $(k^{sym}, k^{mac}) := H(s)$
    else
      $(k^{sym},k^{mac}) \xleftarrow{\$} \mathcal{K}^{sym} \times \mathcal{K}^{mac}$
      Add $(c^{(1)},k^{sym},k^{mac})$ to H_{patch}
    if $\mathsf{Vrfy}_{k^{mac}}(c^{(2)}, c^{(3)}) = 1$
      Return $c_i^{(2)} \oplus k^{sym}$
    else
      Return $\perp$

Procedure FINALIZE(out)
Return $\mathcal{R}((m_i)_{i\in[n]}, \mathfrak{D}, \mathcal{I}, out)$

Fig. 6: Reduction to KEM's OW-PCA security given by the game interface for $\mathcal{A}$.

Proof. Games G_3 and G_4 are identical until ABORTH happens. Given adversary $\mathcal{A}$ run in the REAL-SIM-SO-CCA game, we construct an adversary $\mathcal{B}$ against the OW-PCA security of KEM having success probability $\Pr[\text{ABORTH}]/n$ as depicted in Figure 6. Adversary $\mathcal{B}$ receives a pk and a challenge encapsulation $c^* \leftarrow$ CHALLENGE of some key k^* and aims to output k, given access to an CHECK$(\cdot,\cdot)$ returning $\text{CHECK}_{sk}(k, c) := (\mathsf{Decap}_{sk}(c) \stackrel{?}{=} k)$.

$\mathcal{B}$ runs $\mathcal{A}$ as $\mathcal{A}$ is run in game G_3 except for the following differences: After calling INITIALIZE, $\mathcal{B}$ guesses an index $i^* \xleftarrow{\$} [n]$. $\mathcal{B}$ creates c_i as before, but hides its own

challenge in the first component of the i^{*th} ciphertext. Let's assume that ABORTH happens. Since $\mathcal{B}$ knows $\{c_i^{(1)}\}$ for $i \in [n] \setminus \{i^*\}$, it can detect if $\mathcal{A}$ queries $\text{HASH}(s)$ for $s \in \{k_i\}$ where $i \in [n] \setminus \{i^*\}$, while $\mathcal{B}$ can invoke its CHECK oracle to detect the query $\text{HASH}(k_{i^*})$ since $\text{CHECK}(k_{i^*}, c_{i^*}^{(1)}) = 1$. Therefore $\mathcal{B}$ does not have to guess when ABORTH happens. If $\mathcal{A}$ should call $\text{OPEN}(i^*)$, $\mathcal{B}$ apparently guessed i^* wrong[3] and aborts $\mathcal{A}$. Running the reduction, $\mathcal{B}$ has to maintain the conditions for ABORTDEC. Therefore it suffices to check if $c^{(1)} \in \{c_i^{(1)}\}_{i=1}^n$ and $\mathsf{Vrfy}_{\sigma_i^{mac}}(c^{(2)}, c^{(3)})$ hold, because $H(k)$ cannot be defined, since neither ABORTH, nor ABORT via OPEN happened.

It remains to explain how $\mathcal{B}$ (unable to compute $k = \mathsf{Decap}_{sk}(c^{(1)})$) answers decryption queries without knowing sk. To answer these queries we make use of the nifty "oracle patching technique" from [18]. If $\mathcal{A}$ calls $\text{DEC}(c^{(1)}, c^{(2)}, c^{(3)})$, $\mathcal{B}$ checks if $H(k)$ is already defined by querying $\text{CHECK}(s, c^{(1)})$ for every $(s, \cdot) \in L_H$. If there is such a s, $\mathcal{B}$ uses $(k^{sym}, k^{mac}) := H(s)$. If not, $\mathcal{B}$ picks (k^{sym}, k^{mac}) at random and has to keep an eye on upcoming hash queries, since $\mathcal{B}$ just committed to $H(k)$.

Therefore $\mathcal{B}$ maintains a dedicated list H_{patch} where $\mathcal{B}$ adds $(c^{(1)}, (^{sym}, k^{mac}))$. On every hash query $\text{HASH}(s)$, $\mathcal{B}$ checks if there is an entry $(c^{(1)}, k^{sym}, k^{mac}) \in H_{patch}$ s.t. $\text{CHECK}(s, c^{(1)}) = 1$ in order to fix the oracle by setting $H(s) := (k^{sym}, k^{mac})$. If $\mathcal{A}$ should call $\text{DEC}(c_{i^*}^{(1)}, \cdot, \cdot)$, challenger $\mathcal{C}_3$ treats it like every other decryption query. Considering that ABORTH happens, $\mathcal{B}$ only has to pick the right ciphertext to hide its own OW-PCA challenge to win its game. Therefore $\mathcal{B}$ succeeds if ABORTH happens and $\mathcal{B}$ guessed $i^* \in [n]$ correctly:

$$\mathbf{Adv}_{\mathsf{KEM}}^{\mathsf{OW\text{-}PCA}}(\mathcal{B}, \kappa) \geq \Pr[\text{ABORTH}]/n.$$

Claim 5. There exists a simulator $\mathcal{S}$ run in the IDEAL-SIM-SO-CCA game such that $\mathbf{Adv}(\mathsf{G}_4^{\mathcal{A}}, \mathsf{IDEAL\text{-}SIM\text{-}SO\text{-}CCA}^{\mathcal{S}}_{\mathsf{PKE}_{\mathsf{KEM},\mathsf{MAC}}}) = 0$.

Proof. The simulator runs the adversary as it is run in game G_4, i.e. $\mathcal{S}$ runs PKEGen on its own and feeds pk to $\mathcal{A}$. On $\mathcal{A}$'s call of $\text{ENC}(\mathfrak{D})$ the simulator calls $\text{ENC}(\mathfrak{D})$ as well and creates dummy encryptions without knowing the sampled messages $(m_i)_{i \in [n]}$. If $\mathcal{A}$ calls $\text{OPEN}(i)$, $\mathcal{S}$ forwards the query to its own game, learns m_i, and returns (m_i, r_i) to $\mathcal{A}$.
Because ABORTEARLY does not happen, $\mathcal{S}$ does not have to commit to $\mathsf{Dec}(c_i)$ before ENC is called. Since neither ABORTH nor ABORTDEC happen, $\mathcal{A}$ calls $\text{OPEN}(i)$ before issuing "critical" hash or decryption queries and $\mathcal{S}$ is able to learn m_i and can program H accordingly. Due to these changes and the dummy encryption introduced in game G_2, $\mathcal{A}$ cannot get information on some m_i without calling $\text{OPEN}(i)$, that is, "avowing" $\mathcal{S}$ to call $\text{OPEN}(i)$ as well, allowing $\mathcal{S}$ to answer possibly upcoming hash or decryption queries consistently.

Collecting the advantages of $\mathcal{A}$ we get the claim as stated in Equation (1).

[3] $\mathcal{A}$ cannot ask to open every single challenge ciphertext, since ABORTH occurs.

3.3 Implications for practical encryption schemes

We now give specific instantiations of SIM-SO-CCA secure scheme via our generic transformation. We focus on two well known KEMs, namely the DH and RSA key encapsulation mechanism.

DHIES. Let $\mathbb{G}$ be a group of prime-order p, and let g be a generator. The Diffie-Hellman KEM DH-KEM $= (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is defined as follows. The key-generation algorithm Gen picks $x \xleftarrow{\$} \mathbb{Z}_p$ and defines $pk = X := g^x$ and $sk = x$; the encapsulation algorithm Encap_{pk} picks $r \xleftarrow{\$} \mathbb{Z}_p$ and returns $(c = g^r, k = X^r)$; the decapsulation algorithm $\mathsf{Decap}_{sk}(c)$ returns $k = c^x$. OW-PCA security of the DH-KEM is equivalent to the strong Diffie-Hellman (sDH) assumption [1]. The sDH assumption states that there is no PPT adversary $\mathcal{A}$ that, given two random group elements $U := g^u, V := g^v$ and a *restricted* DDH oracle $\mathcal{O}_v(\cdot,\cdot)$ where $\mathcal{O}_v(a,b) := (a^v \stackrel{?}{=} b)$ computes g^{uv} with non-negligible probability.

Procedure $\mathsf{PKEGen}(1^\kappa)$	**Procedure** $\mathsf{Enc}(m)$	**Procedure** $\mathsf{Dec}(c_1, c_2, c_3)$
$H \xleftarrow{\$} \mathcal{H}$	$r \xleftarrow{\$} \mathbb{Z}_p$	$(k^{sym}, k^{mac}) \leftarrow H({c_1}^x)$
$x \xleftarrow{\$} \mathbb{Z}_p$	$(k^{sym}, k^{mac}) \leftarrow H(X^r)$	if $\mathsf{Vrfy}_{k^{mac}}(c_2, c_3) = 1$
$X := g^x$	$c_1 := g^r$	Return $c_2 \oplus k^{sym}$
$pk := (\mathbb{G}, g, p, X)$	$c_2 := k^{sym} \oplus m$	else
$sk := x$	$c_3 := \mathsf{Tag}_{k^{mac}}(c_2)$	Return $\perp$
Return (pk, sk)	Return (c_1, c_2, c_3)	

Fig. 7: The Diffie-Hellman Integrated Encryption Scheme DHIES instantiated with a one-time pad.

Let MAC be a MAC with message-space and key-space $\{0,1\}^\ell$ and let $\mathcal{H} : \mathbb{G} \mapsto \{0,1\}^{2\ell}$ be a family of hash functions. The security of DHIES $= \mathsf{PKE}_{\mathsf{DH\text{-}KEM},\mathsf{MAC}}$ (depicted in Figure 7)(instantiated with a one-time pad) is stated in the following corollary, whose proof is a direct consequence of Theorem 6.

Corollary 7 DHIES instantiated with a one-time pad is SIM-SO-CCA secure in the random oracle model, if MAC is sUF-OT-CMA and the sDH assumption holds.

RSA-KEM. We obtain another selective-opening secure encryption scheme, if we plug in the RSA-KEM in the generic transformation given in Figure 4. Thereby, OW-PCA security of the RSA-KEM holds under the RSA assumption [42]. Under the RSA assumption, $\mathsf{PKE}_{\mathsf{RSA\text{-}KEM},\mathsf{MAC}}$ (as described in ISO18033-2 [42]) is SIM-SO-CCA secure in the random oracle model.

Both reductions for the OW-PCA security of the DH-KEM, RSA-KEM, respectively, are tight, while both KEMs have unique encapsulations.

4 The OAEP Transformation

In this section we show that OAEP is SIM-SO-CCA secure when instantiated with a *partial-domain one-way trapdoor permutation* (see Section 4.1). Since it is known [22] that the RSA permutation is partial-domain one way under the RSA assumption, this implies that RSA-OAEP is SIM-SO-CCA secure under the RSA assumption. In fact, our result works not only for trapdoor permutations, but for *injective* trapdoor functions as well. Since SIM-SO-CCA security implies IND-CCA security, our proof also provides an alternative to the IND-CCA security proof of [22].

4.1 Trapdoor Permutations and Partial-Domain Onewayness

Recall that a trapdoor permutation is a triple of algorithms $\mathcal{T} = (GK, F, F^{-1})$, where GK generates a key pair $(ek, td) \xleftarrow{\$} GK(1^\kappa)$, $F(ek, \cdot)$ implements a permutation

$$f_{ek} : \{0,1\}^k \to \{0,1\}^k \tag{2}$$

specified by ek, and $F^{-1}(td, \cdot)$ inverts f_{ek} using the trapdoor td. Let us write the function f_{ek} from (2) as a function

$$f_{ek} : \{0,1\}^{\ell+k_1} \times \{0,1\}^{k_0} \to \{0,1\}^k$$

with $k = \ell + k_1 + k_0$.

Definition 8 Let $\mathcal{T}$ be a trapdoor permutation as given above and $\mathcal{B}$ an adversary run in the PD-OW$_\mathcal{T}$ game given in Figure 8. We restrict $\mathcal{B}$ to call CHALLENGE exactly one time and define $\mathcal{B}$'s advantage in winning the PD-OW$_\mathcal{T}$ game as

$$\mathbf{Adv}^{\mathsf{PD\text{-}OW}}_{\mathcal{T}}(\mathcal{B}, \kappa) := \Pr[\mathsf{PD\text{-}OW}^{\mathcal{B}}_{\mathcal{T}} \Rightarrow 1].$$

Moreover, if $\mathbf{Adv}^{\mathsf{PD\text{-}OW}}_{\mathcal{T}}(\mathcal{B}, \kappa) \leq \mathsf{negl}(\kappa)$ for all probabilistic polynomial-time (in κ) adversaries $\mathcal{B}$, we say that $\mathcal{T}$ is a *partial-domain secure* trapdoor permutation.

Procedure INITIALIZE(1^κ)	**Procedure** CHALLENGE	**Procedure** FINALIZE(s')
$(ek, td) \xleftarrow{\$} GK(1^\kappa)$ Return ek	$(s,t) \xleftarrow{\$} \{0,1\}^{\ell+k_1} \times \{0,1\}^{k_0}$ $y := F(ek, (s,t))$ Return y	Return $(s \stackrel{?}{=} s')$

Fig. 8: PD-OW$_\mathcal{T}$ game

4.2 Optimal Asymmetric Encryption Padding (OAEP)

Let $\mathcal{T} = (GK, F, F^{-1})$ be a trapdoor permutation. The OAEP encryption scheme is defined as follows.

- The key generation $\mathsf{Gen}(1^\kappa)$ computes a key pair $(ek, td) \leftarrow GK(1^\kappa)$ for the trapdoor permutation. It defines two hash functions

$$G : \{0,1\}^{k_0} \rightarrow \{0,1\}^{\ell+k_1} \text{ and } H : \{0,1\}^{\ell+k_1} \rightarrow \{0,1\}^{k_0}$$

  and outputs $sk = td$ and $pk = (ek, G, H)$.
- To encrypt a message $m \in \{0,1\}^\ell$, the sender draws a random value $r \xleftarrow{\$} \{0,1\}^{k_0}$. Then it computes

$$s = m||0^{k_1} \oplus G(r) \qquad t = r \oplus H(s).$$

  The ciphertext is $C = F(ek, (s,t)) = f_{ek}(s,t)$.
- To decrypt a ciphertext C, the decryption algorithm $\mathsf{Dec}_{sk}(c)$ uses $sk = td$ to apply the inverse permutation to c, and obtains $(s,t) = F^{-1}(td, c)$. Then it computes $r = t \oplus H(s)$ and $\mu = s \oplus G(r)$, and parses $\mu \in \{0,1\}^{\ell+k_1}$ as $\mu = m||\rho$ with $m \in \{0,1\}^\ell$ and $\rho \in \{0,1\}^{k_1}$. If $\rho = 0^{k_1}$, then the decryption algorithm outputs m. Otherwise $\perp$ is returned.

The OAEP padding process is illustrated in Figure 9.

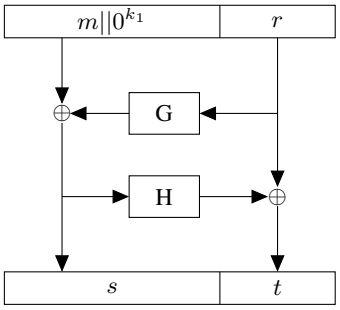

Fig. 9: The OAEP padding process.

4.3 Security of OAEP against SO-CCA Attacks

In this section we will analyze the security of the OAEP scheme. We will prove that OAEP is SIM-SO-CCA-secure in the random oracle model [7], assuming the partial-domain onewayness of the trapdoor permutation $\mathcal{T}$. Note that a proof in the random oracle model is the strongest result we can hope for, since SIM-SO-CCA-security implies IND-CCA security, and it is known [31] that OAEP can not be proven IND-CCA secure without random oracles.

Theorem 9 *Let* OAEP *be the scheme described in Section 4.2 and* $\mathcal{T} = (GK, F, F^{-1})$ *be a trapdoor permutation. Then* OAEP *is* SIM-SO-CCA *secure in the random oracle model (where both hash functions* G *and* H *are modeled as random oracles). In particular, for every PPT relation* $\mathcal{R}$*, every adversary* $\mathcal{A}$ *run in the* $\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA}_{\mathsf{OAEP}}$ *game that issues at most* q_h *queries to* H*,* q_g *queries to* G*,* q_d *decryption queries, and obtains* n *ciphertexts, there exists a simulator* $\mathcal{S}$ *and an adversary* $\mathcal{B}$ *in the* $\mathsf{PD\text{-}OW}_{\mathcal{T}}$ *experiment such that*

$$\mathbf{Adv}^{\mathsf{SIM\text{-}SO\text{-}CCA}}_{\mathsf{OAEP}}(\mathcal{A}, \mathcal{S}, \mathcal{R}, n, \kappa) \leq \delta$$

where

$$\delta = q_d \cdot \left(2^{-k_1} + q_g \cdot 2^{-k_0}\right) + n(q_g + n) \cdot 2^{-k_0} + nq_h \cdot \mathsf{Adv}^{\mathcal{T}}_{\mathsf{pd}}(\mathcal{B}, \kappa) + nq_g \cdot 2^{-\ell - k_1}.$$

Intuition for the proof of Theorem 9. We prove the theorem in a sequence of games, starting with the $\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA}_{\mathsf{OAEP}}$ experiment. From game to game we gradually modify the challenger, until we end up in a game where the challenger can act as a simulator in the $\mathsf{IDEAL\text{-}SIM\text{-}SO\text{-}CCA}_{\mathsf{OAEP}}$ experiment. Our goal is to modify the challenger such that in the final game it does not need to know message m_i before the adversary asks $\textsc{Open}(i)$. To this end, we have to describe how the challenger is able to create "non-committing" ciphertexts $c_1, \ldots, c_n$ in the $\textsc{Enc}$-procedure, which can then be opened to any message m_i when $\mathcal{A}$ issues an $\textsc{Open}(i)$-query.

In a first step, we replace the original decryption procedure that uses the real trapdoor td with an equivalent (up to a negligible error probability) decryption procedure, which is able to decrypt ciphertexts by examining the sequence of random oracle queries made by adversary $\mathcal{A}$. Here we use that $\mathcal{A}$ is not able (except for some non-negligible probability) to create a new valid ciphertext $c = F(ek, (s, t))$, unless it asks the random oracle H on input s and G on input $H(s) \oplus t$. However, in this case the challenger is able to decrypt c by exhaustive search through all queries to H and G made by $\mathcal{A}$.

For $i \in [n]$ let $c_i = F(ek, (s_i, t_i)$ now denote the i^{th} challenge ciphertext that $\mathcal{A}$ receives in the security experiment. We show how to construct an attacker against the partial-domain one-wayness of $\mathcal{T}$, which is successful if the adversary $\mathcal{A}$ ever asks $H(s_i)$ before $\textsc{Open}(i)$ for any $i \in [n]$. Thus, assuming that $\mathcal{T}$ is secure in the sense of partial-domain one-wayness, it will never happen that $\mathcal{A}$ asks $H(s_i)$ before $\textsc{Open}(i)$, except for some negligible probability.

Finally, we conclude with the observation that from $\mathcal{A}$'s point of view all values of $H(s_i)$ remain equally likely until $\textsc{Open}(i)$ is asked, which implies also that it is very unlikely that $\mathcal{A}$ ever asks $G(t_i \oplus H(s_i))$ before $\textsc{Open}(i)$. This in turn means that the challenger does not have to commit to a particular value of $G(t_i \oplus H(s_i))$, and thus not to a particular message $m_i||0^{k_1} = s_i \oplus G(t_i \oplus H(s_i))$, before $\textsc{Open}(i)$ is asked.

Proof of Theorem 9. The proof proceeds in a *sequence of games*, following [9, 43], where Game 0 corresponds to the $\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA}^{\mathcal{A}}_{\mathsf{OAEP}}$-experiment with adversary $\mathcal{A}$ and a challenger, called $\mathcal{C}_0$. From game to game, we gradually modify the challenger, until we obtain a challenger which is able to act as a simulator in the $\mathsf{IDEAL\text{-}SIM\text{-}SO\text{-}CCA}^{\mathcal{S}}_{\mathsf{OAEP}}$ experiment.

Let us first fix some notation. We denote with q_g the number of queries issued by $\mathcal{A}$ to random oracle G, with q_h the number of queries to H, and with q_d the number of decryption queries. For $i \in [n]$ we will denote with c_i the i^{th} component of the challenge ciphertext vector $(c_i)_{i\in[n]}$, and we write c_i as $c_i = f_{ek}(s_i, t_i)$.

Procedure INITIALIZE
$(ek, td) \xleftarrow{\$} GK(1^\kappa)$
Return ek

Procedure ENC($\mathfrak{D}$)
$(m_i)_{i\in[n]} \leftarrow \mathfrak{D}$
for $i \in [n]$:
  $r_i \xleftarrow{\$} \{0,1\}^{k_0}$
  $s_i := m||0^{k_1} \oplus G_{\mathsf{int}}(r_i)$
  $t_i := r_i \oplus H_{\mathsf{int}}(s_i)$
  $c_i := F(ek, (s_i, t_i))$
Return $(c_i)_{i\in[n]}$

Procedure OPEN(i)
$\mathcal{I} := \mathcal{I} \cup \{i\}$
Return (m_i, r_i)

Internal procedure $H_{\mathsf{int}}(s)$
If $(s, h_s) \notin L_H$
  $h_s \xleftarrow{\$} \{0,1\}^{k_0}$
  $L_H := L_H \cup (s, h_s)$
Return h_s

Procedure $H(s)$
$L_H^{\mathcal{A}} := L_H^{\mathcal{A}} \cup \{s\}$
Return $H_{\mathsf{int}}(s)$

Procedure DEC(c)
$(s, t) := F^{-1}(td, c)$
$r := t \oplus H_{\mathsf{int}}(s)$
$m||\rho := s \oplus G_{\mathsf{int}}(r)$
if $\rho = 0^{k_1}$
  Return m
else
  Return $\perp$

Internal procedure $G_{\mathsf{int}}(r)$
if $(r, h_r) \notin L_G$
  $h_r \xleftarrow{\$} \{0,1\}^{\ell+k_1}$
  $L_G := L_G \cup (r, h_r)$
Return h_r

Procedure $G(r)$
$L_G^{\mathcal{A}} := L_G^{\mathcal{A}} \cup \{r\}$
Return $G_{\mathsf{int}}(r)$

Procedure FINALIZE(out)
Return
$\mathcal{R}((m_i)_{i\in[n]}, \mathfrak{D}, \mathcal{I}, out)$

Fig. 10: Procedures of Game 0.

Game 0. Challenger $\mathcal{C}_0$ executes the REAL-SIM-SO-CCA experiment with attacker $\mathcal{A}$ by implementing the procedures described in Figure 10. Note that $\mathcal{C}_0$ also implements procedures to simulate the random oracles G and H. To this end, it maintains four lists

$$L_G \subseteq \{0,1\}^{k_0} \times \{0,1\}^{\ell+k_1} \qquad L_H \subseteq \{0,1\}^{\ell+k_1} \times \{0,1\}^{k_0}$$
$$L_G^{\mathcal{A}} \subseteq \{0,1\}^{k_0} \qquad L_H^{\mathcal{A}} \subseteq \{0,1\}^{\ell+k_1}$$

which are initialized to the empty set in the INITIALIZE procedure.

To simulate the random oracle G, the challenger uses the *internal* procedure G_{int}, which uses list L_G to ensure consistency of random oracle responses. The adversary does not have direct access to procedure G_{int}, but only via procedure G, which stores all values r queried by $\mathcal{A}$ in an additional list $L_G^{\mathcal{A}}$. This allows us to keep track of all values queried by $\mathcal{A}$. Random oracle H is implemented similarly, with procedures H_{int} and H, using list s L_H and $L_H^{\mathcal{A}}$.

By definition we have

$$\mathbf{Adv}(\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA}^{\mathcal{A}}_{\mathsf{OAEP}}, \mathsf{G}_0^{\mathcal{A}}) = 0.$$

In the following games we will replace $\mathcal{C}_0$ with challenger $\mathcal{C}_i$ in Game i. In the last game, we replace the challenger with a simulator.

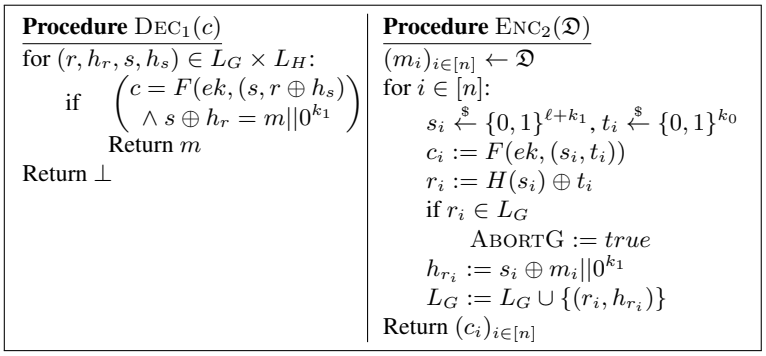

Fig. 11: Replacement procedures DEC_1 and ENC_2.

Game 1. In this game, $\mathcal{C}_1$ proceeds exactly as $\mathcal{C}_0$, except that instead of implementing procedure DEC, it uses procedure DEC_1 from Figure 11 to respond to decryption-queries. Note that procedure DEC_1 does not require the trapdoor td to perform decryption.

Claim 1. It holds that $\mathbf{Adv}(\mathsf{G}_0^{\mathcal{A}}, \mathsf{G}_1^{\mathcal{A}}) \leq q_d \cdot \left(2^{-k_1} + q_g \cdot 2^{-k_0}\right)$.

Proof. Game 1 is perfectly indistinguishable from Game 0, unless $\mathcal{A}$ makes a decryption query with ciphertext c, such that $\mathrm{DEC}(c) \neq \mathrm{DEC}_1(c)$. Note that this can only hold if $\mathcal{A}$ queries a ciphertext c with $(s, t) = F^{-1}(td, c)$, such that

$$(s, \cdot) \notin L_H \quad \text{or} \quad (t \oplus H(s), \cdot) \notin L_G$$

where $\cdot$ is any value, but it holds that $G(t \oplus H(s)) \oplus s = m||\rho$ with $\rho = 0^{k_1}$.

Consider a single chosen-ciphertext $c = F(ek, (s, t))$. Suppose that $(s, \cdot) \notin L_H$. In this case $H(s)$ is uniform and independent from $\mathcal{A}$'s view. The probability that there exists $(r, \cdot) \in L_G$ such that $r = H(s) \oplus t$ is therefore at most $q_g \cdot 2^{-k_0}$, since we assumed that the adversary issues at most q_g queries to G.

If $(r, \cdot) \notin L_G$ then $G(r)$ is uniform and independent from $\mathcal{A}$'s view, thus the probability that $G(r) \oplus s = m||0^{k_1}$ has the correct syntax is at most 2^{-k_1}.

Since the adversary issues at most q_d chosen-ciphertext queries, we have $\mathbf{Adv}(\mathsf{G}_0^{\mathcal{A}}, \mathsf{G}_1^{\mathcal{A}}) \leq q_d \cdot \left(2^{-k_1} + q_g \cdot 2^{-k_0}\right)$.

Game 2. Challenger $\mathcal{C}_2$ proceeds exactly like $\mathcal{C}_1$, except that it implements procedure ENC_2 from Figure 11 instead of ENC. Note that this procedure first samples (s_i, t_i) uniformly random, then computes $c_i = F(ek, (s_i, t_i))$, and finally programs the random oracle G such that c_i decrypts to m_i.

Claim 2. It holds that $\mathbf{Adv}(\mathsf{G}_1^{\mathcal{A}}, \mathsf{G}_2^{\mathcal{A}}) \leq n(q_g + n) \cdot 2^{-k_0}$.

Proof. Note that procedure ENC_2 first defines $r_i := H(s_i) \oplus t_i$ for uniformly random $t_i \xleftarrow{\$} \{0,1\}^{k_0}$. Thus, r_i is distributed uniformly over $\{0,1\}^{k_0}$, exactly as in Game 1.

Now suppose that $r_i \notin L_G$, thus ENC_2 does not terminate. In this case the hash function G is programmed such that $G(r_i) = h_{r_i} = s_i \oplus m_i || 0^{k_1}$. Since s_i is uniformly distributed, so is $G(r_i)$, exactly as in Game 1. Thus, ENC_2 simulates procedure ENC from Game 1 perfectly, provided that it does not terminate.

Note that the procedure terminates only if $r_i \in L_G$. Since all values $r_1, \ldots, r_n$ are distributed uniformly, because the s_i-values are uniformly random, this happens with probability at most $n(q_g + n) \cdot 2^{-k_0}$.

Procedure $\text{OPEN}(i)$	**Procedure** $\text{OPEN}(i)$
$\mathcal{I} := \mathcal{I} \cup \{i\}$	$\mathcal{I} := \mathcal{I} \cup \{i\}$
if $s_i \in L_H^{\mathcal{A}}$	if $s_i \in L_H^{\mathcal{A}}$
$\quad \text{ABORTS} := true$	$\quad \text{ABORTS} := true$
Return (m_i, r_i)	if $r_i \in L_G^{\mathcal{A}}$
	$\quad \text{ABORTR} := true$
	Return (m_i, r_i)

Fig. 12: Modified OPEN-procedures of Games 3 (left) and 4 (right).

Game 3. We add an abort condition to the OPEN-procedure (see the left-hand side of Figure 12). Challenger $\mathcal{C}_3$ proceeds exactly like $\mathcal{C}_2$, except that it raises event ABORTS and terminates, if $\mathcal{A}$ ever queried s_i to H for some $i \in [n]$ *before* querying $\text{OPEN}(i)$.

Note that in Game 3, the attacker never evaluates H on input s_i for any $i \notin \mathcal{I}$, or the game is aborted.

Claim 3. It holds that $\mathbf{Adv}(\mathsf{G}_2^{\mathcal{A}}, \mathsf{G}_3^{\mathcal{A}}) \leq n \cdot q_h \cdot \mathsf{Adv}_{\mathsf{pd}}^{\mathcal{T}}(\mathcal{B}, \kappa)$.

Proof. Game 3 proceeds identically to Game 2, until event ABORTS is raised. Thus we have

$$\mathbf{Adv}(\mathsf{G}_2^{\mathcal{A}}, \mathsf{G}_3^{\mathcal{A}}) \leq \Pr[\text{ABORTS}]$$

We construct an adversary $\mathcal{B}$ against the partial-domain onewayness of $\mathcal{T}$. $\mathcal{B}$ receives as input ek and $y = f_{ek}(s,t)$ for uniformly random $(s,t) \xleftarrow{\$} \{0,1\}^{\ell+k_1} \times \{0,1\}^{k_0}$. It proceeds exactly like $\mathcal{C}_3$, except for the following. At the beginning of the game it sets $pk := ek$ and guesses two indices $j \xleftarrow{\$} [n]$ and $q \xleftarrow{\$} [q_h]$ uniformly random, and sets $c_j := y$. Note that c_j is correctly distributed (cf. the changes introduced in Game 2). When $\mathcal{A}$ makes its q^{th} query s^* to H, then $\mathcal{B}$ returns s^* and terminates.

Assume that ABORTS happens. Then, at some point in the game, $\mathcal{A}$ makes the *first* query s' to H such that $s' = s_i$ is a partial-domain preimage of some c_i. With probability $1/q_h$ it holds that $s^* = s_i$. Moreover, with probability $1/n$ we have $i = j$. In this case $\mathcal{B}$ obtains the partial preimage $s = s_j$ of $y = c_j$. Thus, $\mathcal{B}$ succeeds, if ABORTS happens and if it has guessed $j \in [n]$ and $q \in [q_h]$ correctly. This happens with probability $\Pr[\text{ABORTS}]/(n \cdot q_h)$, which implies that

$$\Pr[\text{ABORTS}] \leq n \cdot q_h \cdot \mathsf{Adv}_{\mathsf{pd}}^{\mathcal{T}}(\mathcal{B}, \kappa).$$

Game 4. We add another abort condition to the Finalize-procedure (see the right-hand side of Figure 12). Challenger $\mathcal{C}_4$ raises event ABORTR and terminates, if $\mathcal{A}$ ever queries r_i to $G_{\mathcal{A}}$ for some $i \in [n]$, before querying $\text{OPEN}(i)$. Otherwise it proceeds like $\mathcal{C}_3$.

Claim 4. It holds that $\mathbf{Adv}(\mathsf{G}_3^{\mathcal{A}}, \mathsf{G}_4^{\mathcal{A}}) \leq n \cdot q_g \cdot 2^{-\ell - k_1}$.

Proof. Note that $\mathcal{A}$ never queries s_i before querying $\text{OPEN}(i)$ (or the game is aborted), due to the changes introduced in Game 3. Thus, for all $i \notin \mathcal{I}$, $H(s_i)$ is uniformly random and independent of $\mathcal{A}$'s view. Therefore, all $r_i = t_i \oplus H(s_i)$ are uniformly random and independent of $\mathcal{A}$'s view. Since $\mathcal{A}$ issues at most q_g queries to G, and we have $1 \leq i \leq n$, this implies $\mathbf{Adv}(\mathsf{G}_3^{\mathcal{A}}, \mathsf{G}_4^{\mathcal{A}}) \leq n \cdot q_g \cdot 2^{-\ell - k_1}$.

Game 5. Note that the attacker in Game 4 never issues a query $G(r_i)$ before asking $\text{OPEN}(i)$, as otherwise the game is aborted. Thus, the challenger does not have to define the hash value $G(r_i)$ before $\text{OPEN}(i)$ is asked. Therefore we can move the definition of $G(r_i)$ from the ENC_2-procedure to the OPEN-procedure.

Therefore we replace the procedures ENC_2 and OPEN from Game 4 with procedures ENC and OPEN described in Figure 13. Note that the only difference is that for each $i \in [n]$ the hash value $G(r_i)$ is not defined in the ENC-procedure, but in the OPEN procedure. Moreover, this modification is completely oblivious to $\mathcal{A}$, which implies

$$\mathbf{Adv}(\mathsf{G}_4^{\mathcal{A}}, \mathsf{G}_5^{\mathcal{A}}) = 0.$$

Procedure $\text{ENC}(\mathfrak{D})$	**Procedure** $\text{OPEN}(i)$
$(m_i)_{i\in[n]} \leftarrow \mathfrak{D}$	$\mathcal{I} := \mathcal{I} \cup \{i\}$
For $i \in [n]$:	if $s_i \in L_H^{\mathcal{A}}$
$\quad s_i \xleftarrow{\$} \{0,1\}^{\ell+k_1}, t_i \xleftarrow{\$} \{0,1\}^{k_0}$	$\quad$ ABORTS $:= true$
$\quad c_i := F(ek, (s_i, t_i))$	if $r_i \in L_G^{\mathcal{A}}$
$\quad r_i := H(s_i) \oplus t_i$	$\quad$ ABORTR $:= true$
$\quad$ if $r_i \in L_G$	$h_{r_i} := s_i \oplus m_i \| 0^{k_1}$
$\quad\quad$ ABORTG $:= true$	$L_G := L_G \cup \{(r_i, h_{r_i})\}$
Return $(c_i)_{i\in[n]}$	Return (m_i, r_i)

Fig. 13: New procedures for Game 5.

Game 6. Note that in Game 5 the encryption procedure samples a message vector $(m_i)_{i\in[n]}$, but the messages are only used in the OPEN-procedure. This allows us to construct a simulator, whose procedures are described in Figure 14. Note that the view of $\mathcal{A}$ when interacting with the simulator is *identical* to its view when interacting with challenger $\mathcal{C}_5$, which implies

$$\mathbf{Adv}(\mathsf{G}_5^{\mathcal{A}}, \mathsf{G}_6^{\mathcal{A}}) = 0$$

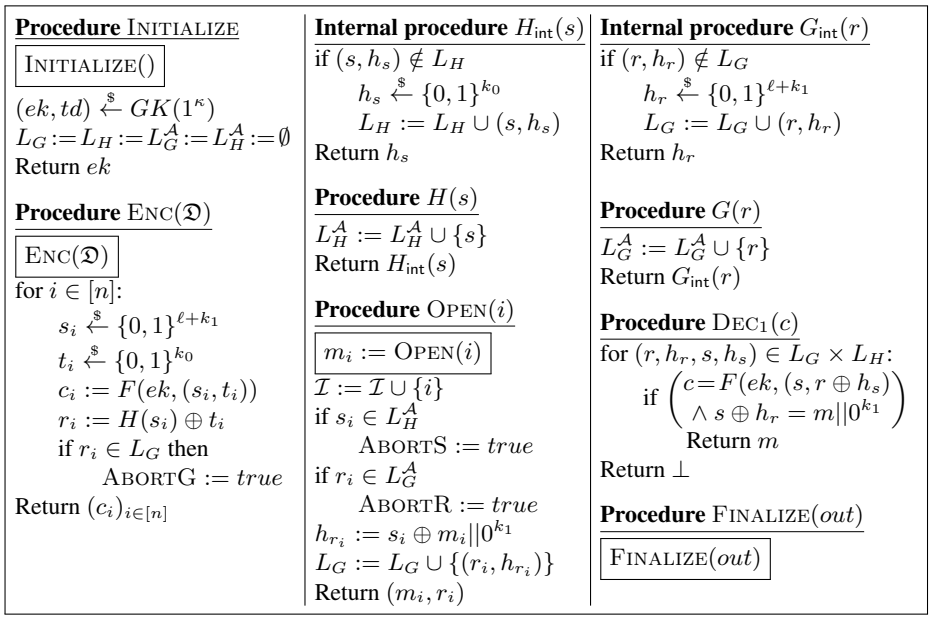

Fig. 14: Procedures used by the simulator to implement the $\mathsf{REAL\text{-}SIM\text{-}SO\text{-}CCA}^{\mathcal{A}}_{\mathsf{OAEP}}$ experiment. Instructions in boxes correspond to calls to the $\mathsf{IDEAL\text{-}SIM\text{-}SO\text{-}CCA}^{\mathcal{S}}_{\mathsf{OAEP}}$-experiment made by the simulator.

Acknowledgements

We thank Zhengan Huang and Shengli Liu for their valuable comments. Felix Heuer and Eike Kiltz were (partially) funded by a Sofja Kovalevskaja Award of the Alexander von Humboldt Foundation and the German Federal Ministry for Education and Research. Felix Heuer was also partially funded by the German Israeli Foundation. Sven Schäge is supported by Ubicrypt, the research training group 1817/1 funded by the German Research Foundation (DFG). Part of this work was done while he was employed at University College London and supported by EPSRC grant EP/J009520/1.

References

1. M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In Naccache [33], pages 143–158.
2. M. Backes, M. Dürmuth, and D. Unruh. OAEP is secure under key-dependent messages. In J. Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 506–523, Melbourne, Australia, Dec. 7–11, 2008. Springer, Berlin, Germany.
3. D. Beaver. Plug and play encryption. In Kaliski Jr. [29], pages 75–89.
4. D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In R. A. Rueppel, editor, *EUROCRYPT'92*, volume 658 of *LNCS*, pages 307–323, Balatonfüred, Hungary, May 24–28, 1992. Springer, Berlin, Germany.

5. M. Bellare, R. Dowsley, B. Waters, and S. Yilek. Standard security does not imply security against selective-opening. In Pointcheval and Johansson [37], pages 645–662.
6. M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Joux [28], pages 1–35.
7. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73, Fairfax, Virginia, USA, Nov. 3–5, 1993. ACM Press.
8. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In A. D. Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 92–111, Perugia, Italy, May 9–12, 1994. Springer, Berlin, Germany.
9. M. Bellare and P. Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In S. Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Germany.
10. M. Bellare, B. Waters, and S. Yilek. Identity-based encryption secure against selective opening attack. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 235–252, Providence, RI, USA, Mar. 28–30, 2011. Springer, Berlin, Germany.
11. 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, Darmstadt, Germany, May 21–23, 2012. Springer, Berlin, Germany.
12. A. Boldyreva and M. Fischlin. On the security of OAEP. In X. Lai and K. Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 210–225, Shanghai, China, Dec. 3–7, 2006. Springer, Berlin, Germany.
13. D. R. L. Brown. What hashes make RSA-OAEP secure? Cryptology ePrint Archive, Report 2006/223, 2006. `http://eprint.iacr.org/`.
14. R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In Kaliski Jr. [29], pages 90–104.
15. R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648, Philadephia, Pennsylvania, USA, May 22–24, 1996. ACM Press.
16. R. Canetti, S. Halevi, and J. Katz. Adaptively-secure, non-interactive public-key encryption. In J. Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 150–168, Cambridge, MA, USA, Feb. 10–12, 2005. Springer, Berlin, Germany.
17. T. Clancy and W. Arbaugh. Extensible Authentication Protocol (EAP) Password Authenticated Exchange. RFC 4746 (Informational), Nov. 2006.
18. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2003.
19. T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008. Updated by RFCs 5746, 5878, 6176.
20. 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, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.
21. E. Fujisaki. All-but-many encryptions: A new framework for fully-equipped UC commitments. Cryptology ePrint Archive, Report 2012/379, 2012. `http://eprint.iacr.org/`.
22. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 260–274, Santa Barbara, CA, USA, Aug. 19–23, 2001. Springer, Berlin, Germany.
23. B. Harris. RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol. RFC 4432 (Proposed Standard), Mar. 2006.

24. B. Hemenway, B. Libert, R. Ostrovsky, and D. Vergnaud. Lossy encryption: Constructions from general assumptions and efficient selective opening chosen ciphertext security. In D. H. Lee and X. Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 70–88, Seoul, South Korea, Dec. 4–8, 2011. Springer, Berlin, Germany.
25. D. Hofheinz. All-but-many lossy trapdoor functions. In Pointcheval and Johansson [37], pages 209–227.
26. 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, San Diego, CA, USA, Feb. 24–26, 2014. Springer, Berlin, Germany.
27. R. Housley. Use of the RSAES-OAEP Key Transport Algorithm in Cryptographic Message Syntax (CMS). RFC 3560 (Proposed Standard), July 2003.
28. A. Joux, editor. *EUROCRYPT 2009*, volume 5479 of *LNCS*, Cologne, Germany, Apr. 26–30, 2009. Springer, Berlin, Germany.
29. B. S. Kaliski Jr., editor. *CRYPTO'97*, volume 1294 of *LNCS*, Santa Barbara, CA, USA, Aug. 17–21, 1997. Springer, Berlin, Germany.
30. E. Kiltz, A. O'Neill, and A. Smith. Instantiability of RSA-OAEP under chosen-plaintext attack. In T. Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 295–313, Santa Barbara, CA, USA, Aug. 15–19, 2010. Springer, Berlin, Germany.
31. E. Kiltz and K. Pietrzak. On the security of padding-based encryption schemes - or - why we cannot prove OAEP secure in the standard model. In Joux [28], pages 389–406.
32. J. Lai, R. H. Deng, S. Liu, J. Weng, and Y. Zhao. Identity-based encryption secure against selective opening chosen-ciphertext attack. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 77–92, Copenhagen, Denmark, May 11–15, 2014. Springer, Berlin, Germany.
33. D. Naccache, editor. *CT-RSA 2001*, volume 2020 of *LNCS*, San Francisco, CA, USA, Apr. 8–12, 2001. Springer, Berlin, Germany.
34. T. Nadeau, C. Srinivasan, and A. Farrel. Multiprotocol Label Switching (MPLS) Management Overview. RFC 4221 (Informational), Nov. 2005.
35. T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In Naccache [33], pages 159–175.
36. 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, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.
37. D. Pointcheval and T. Johansson, editors. *EUROCRYPT 2012*, volume 7237 of *LNCS*, Cambridge, UK, Apr. 15–19, 2012. Springer, Berlin, Germany.
38. K. Raeburn. Encryption and Checksum Specifications for Kerberos 5. RFC 3961 (Proposed Standard), Feb. 2005.
39. B. Ramsdell and S. Turner. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC 5751 (Proposed Standard), Jan. 2010.
40. E. Rescorla. Preventing the Million Message Attack on Cryptographic Message Syntax. RFC 3218 (Informational), Jan. 2002.
41. V. Shoup. OAEP reconsidered. *Journal of Cryptology*, 15(4):223–249, 2002.
42. V. Shoup. ISO 18033-2: An emerging standard for public-key encryption. `http://shoup.net/iso/std6.pdf`, Dec. 2004. Final Committee Draft.
43. V. Shoup. Sequences of games: a tool for taming complexity in security proofs, 2004. shoup@cs.nyu.edu 13166 received 30 Nov 2004, last revised 18 Jan 2006.
44. R. Steinfeld, J. Baek, and Y. Zheng. On the necessity of strong assumptions for the security of a class of asymmetric encryption schemes. In L. M. Batten and J. Seberry, editors, *ACISP 02*, volume 2384 of *LNCS*, pages 241–256, Melbourne, Victoria, Australia, July 3–5, 2002. Springer, Berlin, Germany.