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 $c_i = \text{Enc}_{pk_S}(m_i; r_i)$ to $S$. Assume an adversary observes these ciphertexts, and is then able to “corrupt” a subset of clients $\{A_i\}_{i \in I}, I \subseteq \{1, \ldots, n\}$, for instance by installing a malware on their computers. Then, for all $i \in I$, 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 and full IND-SO security are the strongest notions of security [11, 5, 26]. However, only SIM-SO 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 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 \( \mu, \nu \), we denote with \( \mu || \nu \) the string obtained by concatenating \( \mu \) with \( \nu \). 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 \( \kappa \) 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 \( \kappa \)). For a set \( A \) we denote the sampling of a uniform random element \( a \) by \( a \overset{\$}{\leftarrow} A \), while we denote the sampling according to some distribution \( D \) by \( a \overset{\$}{\leftarrow} D \).

### 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}, P_1, P_2, \ldots, P_t, \text{Finalize}\} for \( t \geq 0 \), where \( P_1 \) to \( P_t \) and \text{Finalize} might require some input parameters, while \text{Initialize} is run on the security parameter \( \kappa \). 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 \( \epsilon \). An adversary \( A \) is run in game \( G \) (by challenger \( C \)), if \( A \) calls \text{Initialize}. During the game \( A \) may run the procedures \( P_i \) as often as allowed by the game. If a procedure \( P \) was called by \( A \), the output of \( P \) is returned to \( A \), except for the \text{Finalize} procedure. On \( A \)’s call of \text{Finalize} the game ends and outputs whatever \text{Finalize} returns. The output \( \text{out} \) of a game \( G \) that runs \( A \) is denoted as \( G^A \Rightarrow \text{out} \). If a game’s output is either 0 or 1, \( A \) wins \( G \) if \( G^A \Rightarrow 1 \). Further, the advantage \( \text{Adv}(G^A, H^A) \) of \( A \) in distinguishing games \( G \) and \( H \) is defined as \( |\Pr[G^A \Rightarrow 1] – \Pr[H^A \Rightarrow 1]| \). For \( A \) run in \( G \) and \( S \) run in game \( H \) the advantage of \( A \) is defined as \( |\Pr[G^A \Rightarrow 1] – \Pr[H^S \Rightarrow 1]| \). Setting a boolean flag “\text{Abort...}” to \( \text{true} \) implicitly aborts the adversary.

### 2.2 Public Key Encryption Schemes

Let \( \mathcal{M}, \mathcal{R}, \mathcal{C} \) be sets. We say that \( \mathcal{M} \) is the message space, \( \mathcal{R} \) is the randomness space, and \( \mathcal{C} \) is the ciphertext space. A public key encryption scheme \( \text{PKE} = (\text{PKEGen}, \text{Enc}, \text{Dec}) \) consists of three polynomial-time algorithms.
On the Selective Opening Security of Practical Public-Key Encryption Schemes

– Gen generates, given the unary representation of the security parameter $1^κ$, a key pair $(sk, pk) ←$ Gen$(1^κ)$, where $pk$ defines $\mathcal{M}$, $\mathcal{R}$, and $\mathcal{C}$.

– Given $pk$, and a message $m ∈ \mathcal{M}$ Enc outputs an encryption $c ← Enc_{pk}(m) ∈ \mathcal{C}$ of $m$ under the public key $pk$.

– The decryption algorithm Dec takes a secret key $sk$ and a ciphertext $c ∈ \mathcal{C}$ as input, and outputs a message $m = Dec_{sk}(c) ∈ \mathcal{M}$, or a special symbol ⊥$∉$ $\mathcal{M}$ 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 $c = Enc(m; r)$ for $r ← \mathcal{R}$. We require the PKE to be correct, that is for all security parameters $1^κ$, for all $(pk, sk) ←$ PKEGen$(1^κ)$, and for all $m ∈ \mathcal{M}$ we have $Pr[Dec_{sk}(Enc_{pk}(m)) = m] = 1$

2.3 SIM-SO-CCA Security Definition

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

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

Fig. 2: IDEAL-SIM-SO-CCA$_{PKE}$ game.
Further, \( A \) is not allowed to call \( \text{Dec} \) on any \( c_i \). To an adversary \( A \), a simulator \( S \), a relation \( \mathcal{R} \) and \( n \) we associate the advantage function

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

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

### 3 Transformation from any OW-PCA secure KEM

#### 3.1 Key Encapsulation Mechanisms and Message Authentication Codes

**Definition 2** Let \( \mathcal{R} \) a key space, \( \mathcal{R} \) a randomness space, and \( \mathcal{C} \) a ciphertext space.

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

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

We require correctness: for all \( \kappa \in \mathbb{N} \), for all \( (pk, sk) \) generated by \( \text{KEMGen}(1^\kappa) \), and for all \( (c, k) \) output by \( \text{Encap}_{pk} \) we have \( \Pr[\text{Decap}_{sk}(c) \neq k] = 0 \). We make the randomness used in \( \text{Encap} \) only explicit when needed. Without loss of generality we assume \( \text{Encap} \) to sample \( k \in \mathcal{R} \), and \( \mathcal{R}, \mathcal{C} \) to be exponentially large in the security parameter: \( |\mathcal{R}| \geq 2^n, |\mathcal{C}| \geq 2^n \).

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

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 \( \text{CHECK}_{sk}(c, k) := (\text{Decap}_{sk}(c) \rightleftharpoons k) \in \{0, 1\} \). Since an indistinguishability based security notion is out of reach, once \( A \) is granted access to \( \text{CHECK} \), we make use of a weaker security notion, given in the following definition.

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

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

A KEM is OW-PCA secure, if \( \text{Adv}^{\text{OW-PCA}}_{\text{KEM}}(A, \kappa) \) is negligible for all PPT \( A \).
Fig. 3: OW-PCA\textsubscript{KEM} game

Definition 4 Let $\mathcal{M}$ 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 $\text{MAC} = (\text{MACGen}, \text{Tag}, \text{Vrfy})$, whereby

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

We require $\text{MAC}$ to be correct: For all $\kappa \in \mathbb{N}$, all keys $k$ generated by $\text{MACGen}(1^\kappa)$, all $m \in \mathcal{M}$ and all tags computed by $\text{Tag}_k(m)$ we have $\Pr[\text{Vrfy}_k(m, \text{Tag}_k(m)) = 1] = 1$.

For a fixed $\text{MAC}$ and $k$, given message $m$ we call a tag $t$ / the tuple $(m,t)$ valid, if $\text{Vrfy}_k(m,t) = 1$.

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

We define the advantage of $\mathcal{A}$ run in the suF-OT-CMA\textsubscript{MAC} game as

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

$\text{MAC}$ is suF-OT-CMA secure, if $\text{Adv}_{\text{MAC}}^{\text{suF-OT-CMA}}(\mathcal{A}, \kappa) \leq \negl(\kappa)$ 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 KEM = (KEMGen, Encap, Decap) be a KEM, H a family of hash functions, and let MAC = (MACGen, Tag, Vrfy) be a MAC. The public-key encryption scheme PKE_{KEM,MAC} obtained by the transformation is given in Figure 4.

<table>
<thead>
<tr>
<th>Procedure PKEGen(1^n)</th>
<th>Procedure ENC(m)</th>
<th>Procedure DEC(c_1, c_2, c_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(pk_{KEM}, sk_{KEM}) \xleftarrow{$} KEMGen(1^n)</td>
<td>(k, c_1) \xleftarrow{$} Encap_{KEM}</td>
<td>k \leftarrow \text{Decap}_{KEM}(c_1)</td>
</tr>
<tr>
<td>H \xleftarrow{$} H</td>
<td>(k_{sym}, k_{mac}) := H(k)</td>
<td>\text{if Vrfy}<em>{k</em>{mac}}(c_2, c_3) = 1</td>
</tr>
<tr>
<td>pk := (pk_{KEM}, H)</td>
<td>c_2 := k_{sym} \oplus m</td>
<td>\text{Return } c_2 \oplus k_{sym}</td>
</tr>
<tr>
<td>sk := sk_{KEM}</td>
<td>c_3 := \text{Tag}<em>{k</em>{mac}}(c_2)</td>
<td>\text{else}</td>
</tr>
<tr>
<td>Return pk</td>
<td>Return (c_1, c_2, c_3)</td>
<td>Return ⊥</td>
</tr>
</tbody>
</table>

Fig. 4: Transformation PKE_{KEM,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 PKE_{KEM,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 PKE_{KEM,MAC} is SIM-SO-CCA secure in the random oracle model. In particular, for any adversary A run in the REAL-SIM-SO-CCA_{PKE_{KEM,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 R, there exists a simulator S, a forger F run in the SUF-OT-CMAC game, and an adversary B run in the OW-PCA_{KEM} game with roughly the same running time as A such that

\[
\text{Adv}_{\text{PKE}_{KEM,MAC}}(A, S, R, n, \kappa) \leq n \cdot \left( q_h + \frac{q_d}{2^{\kappa - 1}} + \text{Adv}_{\text{SUF-OT-CMA}}(F, \kappa) + \text{Adv}_{\text{OW-PCA}}(B, \kappa) \right). \tag{1}
\]

Let us have a high-level look at our proof. Up to some small syntactical changes G₀ constitutes of the REAL-SIM-SO-CCA_{PKE_{KEM,MAC}} game.

The later simulator S will provide A with message-independent dummy encryptions cᵢ. This allows S to claim that cᵢ = Encᵢ₀(mᵢ, rᵢ) for some arbitrary mᵢ after sending cᵢ to A. If A should decide to open cᵢ, Game G₂ introduces the dummy encryptions, while G₁ serves as a preparational step.

After calling ENC(2), A is allowed to make OPEN, HASH and DEC queries in an arbitrary order. Assume, that A did not¹ query OPEN(i) before calling HASH(kᵢ) or issuing a valid decryption query (cᵢ⁽¹⁾, · · ·). Since the indexset of opened messages I is part of A’s output S wants to simulate, S may not query OPEN(i) if A did not make the same call. Neither can S answer such a query, since it would fix kᵢ^{sym} and thereby

¹ Neither HASH(kᵢ) nor DEC(cᵢ⁽¹⁾, · · ·) queries are a tripping hazard, once A called OPEN(i).
$m_i$ before $A$ made a potential OPEN($i$) query. Therefore, we need to block \( \text{Hash}(k_i) \) and valid \( \text{Dec}(c_i^{(1)}, \cdot, \cdot) \) queries, if $A$ did not call 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 $A$’s hash queries $\text{Hash}(k_i)$ 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 $A$, let $n = n(\kappa)$ be a polynomial in $\kappa$. For $i \in [n]$ let: $m_i$ denote the $i^{\text{th}}$ message sampled by the challenger, $r_i$ the $i^{\text{th}}$ randomness used by Encap: $(k_i, c_i^{(1)}) \leftarrow \text{Encap}(r_i)$, $(k_i^{\text{sym}}, k_i^{\text{mac}}) \leftarrow H(k_i)$ the $i^{\text{th}}$ key-pair generated by hashing $k_i$ and $c_i := (c_i^{(1)}, c_i^{(2)}, c_i^{(3)})$ the $i^{\text{th}}$ ciphertext. Without loss of generality, the games samples $(r_i)_{i \in [n]}$ as part of \text{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.](image-url)
Game 0. We model \( H \) as a random oracle. Challenger \( C_0 \) keeps track of issued calls (either by the game or \( A \)) of \( \text{Hash}(s) \) by maintaining a list \( L_H \). For a query \( s \), \( \text{Hash}(s) \) returns \( h_s \) if there is an entry \((s, h_s) \in L_H\), otherwise \( \text{Hash} \) 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 \( C_0 \) samples \((k_i^{\text{sym}}, k_i^{\text{mac}})_{i \in [n]}\) uniformly random and sets \((H(k_i))_{i \in [n]} := (k_i^{\text{sym}}, k_i^{\text{mac}})\) while \text{INITIALIZE} is run. Additionally, \( G_0 \) runs \( \text{Encap}_{pk} \) to generate \((k_i, c_i^{(1)})_{i \in [n]}\) during \text{INITIALIZE}.

Claim 1. \( \text{Adv}(\text{REAL-SIM-SO-CCA}_{\text{PKEM,MAC}}^A, G_0^d) = 0 \).

Proof. Apparently, it makes no difference if the challenger samples \((r_i)_{i \in [n]}\) and runs \( \text{Encap}(r_i) \) on demand as part of \( \text{Enc} \) or in advance while \text{INITIALIZE} is run.

Since \( H \) is modeled as a random oracle, \( H(s) \) is sampled uniformly random for every fresh query \( \text{Hash}(s) \). Therefore \( C_0 \) does not change the distribution by sampling \((k_i^{\text{sym}}, k_i^{\text{mac}})_{i \in [n]}\) in the first place and setting \( H(k_i) := (k_i^{\text{sym}}, k_i^{\text{mac}}) \) afterwards.

Game 1. We add an abort condition. Challenger \( C_1 \) raises the event \text{ABORT EARLY} and aborts \(^2 A \), if \( A \) did not call \( \text{Enc} \) before calling either \( H(k_i) \) or \( \text{Dec}(c_i^{(1)}, \cdot, \cdot) \) for some \( i \in [n] \).

Claim 1. \( \text{Adv}(\text{Encap}^A_{\text{PKEM,MAC}}, G_0^d) \leq n \cdot (q_h + q_d) \cdot 2^{-(\kappa - 1)} \).

Proof. Since games \( G_0 \) and \( G_1 \) are identical until \text{ABORT EARLY} is raised, it follows that \( \text{Adv}(\text{Encap}^A_{\text{PKEM,MAC}}, G_1^d) \leq \text{Pr}[\text{ABORT EARLY}] \). Let \( \text{viaHash} \) and \( \text{viaDec} \) be the events that \text{ABORT EARLY} was caused by either a hash or a decryption query of \( A \). 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 \( A \). It holds that
\[
\text{Pr}[\text{ABORT EARLY}] = \text{Pr}[\text{viaHash}] + \text{Pr}[\text{viaDec}]
\leq \text{Pr}[s_1 \in \{k_i\}_{i=1}^n] + \sum_{i=2}^{q_h} \text{Pr}[s_i \in \{k_i\}_{i=1}^n] \bigwedge_{j=1}^{i-1} s_j \notin \{k_i\}_{i=1}^n] + \sum_{i=2}^{q_d} \text{Pr}[d_i^{(1)} \in \{c_i^{(1)}\}_{i=1}^n] \bigwedge_{j=1}^{i-1} d_j^{(1)} \notin \{c_i^{(1)}\}_{i=1}^n] = \sum_{i=1}^{q_h} \frac{n}{2^{\kappa} - (i-1)} + \sum_{i=1}^{q_d} \frac{n}{2^{\kappa} - (i-1)} \leq \sum_{i=1}^{q_h} \frac{n}{2^{\kappa} - q_h} + \sum_{i=1}^{q_d} \frac{n}{2^{\kappa} - q_d} \leq n(q_h + q_d) \cdot 2^{1-\kappa}.
\]
Above holds since Encap samples \( k \) and KEM has unique encapsulations.

Game 2. We change the encryption procedure and answer hash queries in a different way. \( C_2 \) does not program \( H(k_i) \) for \( i \in [n] \) anymore. \( \text{Enc} \) still samples \( m_i \), and samples \( \sigma_i^{\text{sym}} \sim \mathcal{K}_{\text{sym}}, \sigma_i^{\text{mac}} \sim \mathcal{K}_{\text{mac}} \), to compute \( c_i = \text{Enc}_{k_i}(m_i) := (c_i^{(1)}, \sigma_i^{\text{sym}}, \sigma_i^{\text{mac}}) \).

\(^2\) Notice, that \( C_1 \) aborts even if such a decryption query is invalid.
distribution of the encryptions. If \( A \) adds another abort condition. If \( \sigma \) calls \( \text{Hash}(k_i) \) for \( i \in [n] \) or \( \text{Open}(i) \), 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 \( \text{Open}(i) \) was not called.

Claim 2. \( \text{Adv}(G_1^3, G_2^3) = 0. \)

Proof. Assuming that \( \text{ABORT EARLY} \) does not happen in game \( G_2 \), the keys \( k_i^{sym} \) and \( k_i^{mac} \) are still uniformly random when \( A \) calls \( \text{Enc} \). 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 \( C_2 \) can sample \((c_i^{(2)})_{i \in [n]} := \sigma_i^{sym}\) uniformly and can compute the tags using a uniform key \( \sigma_i^{mac} \) without changing the distribution of the encryptions \((c_i)_{i \in [n]} \).

\( C_2 \) does not program \( H(k_i) \) for \( i \in [n] \) anymore, but has to keep \( H \) consistent. If \( A \) calls \( \text{Hash}(k_i) \) or \( \text{Open}(i) \), \( C_2 \) sets \( H(k_i) := (\sigma_i^{sym} \oplus m_i, \sigma_i^{mac}) \).

Game 3. We add another abort condition. If \( A \) already called \( \text{Enc} \), issues a decryption \( (c_1^{(1)}, c_2^{(2)}, c_3^{(3)}) \notin \{c_i\}_{i \in [n]} \), where \( H(k_i) \) is not defined, and \( \text{Vrfy}_{\sigma_i^{mac}}(c_1^{(1)}, c_2^{(2)}, c_3^{(3)}) \) verifies, challenger \( C_3 \) raises \( \text{ABORT DEC} \) and aborts \( A \).

Claim 3. \( \text{Adv}(G_2^3, G_3^3) \leq n \cdot \text{Adv}_{\text{MAC}}^{\text{UF-OT-CMA}}(F, \kappa). \)

Proof. Games \( G_2 \) and \( G_3 \) are identical until \( \text{ABORT DEC} \) happens, it suffices to bound \( \text{Pr}[\text{ABORT DEC}]. \)

Let \( \text{MAC} := (\text{MACGen}, \text{Tag}, \text{Vrfy}) \) be the MAC used by the sUF-OT-CMA challenger. We construct an adversary \( F \) against the sUF-OT-CMA security of \( \text{MAC} \) having success probability \( \text{Pr}[\text{ABORT DEC}]/n \). The reduction is straight forward: \( F \) runs adversary \( A \) as in game \( G_3 \), but picks \( i^* \overset{\$}{\leftarrow} [n] \) during \text{INITIALIZE}. Computing the \( i^* \text{th} \) ciphertext, \( F \) queries its sUF-OT-CMA challenger for \( t^* := \text{TAG}(\sigma_i^{sym}) \) instead of using its own \text{TAG} procedure and sends \((c_i)_{i \in [n]} \) to \( A \). If \( A \) should call \( \text{OPEN}(t^*) \), challenger \( C_3 \) apparently was unlucky in hiding its own challenge and aborts the adversary. Querying its \( \text{VRFY}_{\kappa}(\cdot, \cdot) \) oracle, \( F \) can detect when \( A \) issues a valid query \( \text{DECE}(c_1^{(1)}, c_2^{(2)}, c_3^{(3)}) \) for some \( i \in [n] \), returns \((c_2^{(2)}, c_3^{(3)}) \) to its sUF-OT-CMA challenger and aborts \( A \).

Assume that \( \text{ABORT DEC} \) happens, i.e. \( A \) makes a valid decryption query \((c_1^{(1)}, c_2^{(2)}, c_3^{(3)}) \notin \{c_i\}_{i \in [n]} \), while \( H(k_i) \) is still undetermined. Notice, that we must not allow \( H(k_i, \cdot) \) to be fixed since \( k_i^{mac} \) is only known to the sUF-OT-CMA challenger. Let \((c_1^{(1)}, c_2^{(2)}, c_3^{(3)}) \in \{c_i\}_{i \in [n]} \) be the ciphertext \( c_i \), whose first component matches the first entry of \( A \)’s valid decryption query. Hence, \( c_3^{(3)} \) is either a new valid tag for \( c_2^{(2)} \) or \( c_3^{(3)} \) is a valid tag for a “new” message \( c_2^{(2)} \), since \((c_2^{(2)}, c_3^{(3)}) \neq (c_2^{(2)}, c_3^{(3)}) \). In both cases \( F \) wins its sUF-OT-CMA challenge by returning \((c_2^{(2)}, c_3^{(3)}) \), if \( F \) picks the right challenge ciphertext to embed \( t^* \). The claim follows by rearranging

\[ \text{Adv}_{\text{MAC}}^{\text{UF-OT-CMA}}(F, \kappa) \geq \text{Pr}[\text{ABORT DEC}]/n. \]

Game 4. We add one more abort condition. Challenger \( C_4 \) raises the event \( \text{ABORT H} \) if \( A \) already called \( \text{Enc} \), issues a hash query \( \text{Hash}(k_i) \) for \( i \in [n] \) and did not call \( \text{OPEN}(i) \) before.

Claim 4. \( \text{Adv}(G_4^3, G_4^3) \leq n \cdot \text{Adv}_{\text{KEM}}^{\text{OW-PCA}}(B, \kappa). \)
Procedure `Initialize`

\[ pk \overset{\$}{\leftarrow} \text{Initialize}_{GW-PCA} \]
\[ i^* \overset{\$}{\leftarrow} [n] \]
\[ (r_i)_{i \in [n]} \overset{\$}{\leftarrow} \mathcal{R} \]
\[ (k_i, c_i)_{i \in [n]} \overset{\$}{\leftarrow} \text{Encap}_{pk}(r_i) \]
\[ c_i \overset{\$}{\leftarrow} \text{CHALLENGE} \]
\[ \text{Return } pk \]

Procedure `Hash(s)`

if \((s, \cdot) \notin L_H\)
  if \(s = k_i\) for some \(i \in [n] \setminus \{i^*\}\)
    if `calledEnc`
      `ABORT`: \(true\)
    else
      `ABORT`: \(true\)
  else
    if `Check(s, c_i^{(1)}) = 1`
      `FINALIZE_{GW-PCA}(s)`
    else
      \((\exists c_i \overset{\$}{\leftarrow} k_{sym, k_{mac}} \in H_{patch}) \text{ s.t. } \text{Check}(s, c_i^{(1)}) = 1\)
      \(H(s) := (k_{sym, k_{mac}})\)
    else
      \(h_s \overset{\$}{\leftarrow} k_{sym} \times k_{mac}\)
      \(H(s) := h_s\)
\[ \text{Return } H(s) \]

Procedure `Open(i)`

if \(i = i^*\)
  `ABORT`: \(true\)
else
  \(I := I \cup \{i\}\)
  \(H(k_i) := (\sigma_i^{sym} \oplus m_i, \sigma_i^{mac})\)
\[ \text{Return } (m_i, r_i) \]

Procedure `Enc(D)`

\(\text{calledEnc} := \text{true} \)
\[
(m_i)_{i \in [n]} \leftarrow D.
\]
\[
(\sigma_i^{sym}, \sigma_i^{mac})_{i \in [n]} \overset{\$}{\leftarrow} k_{sym} \times k_{mac}.
\]
\[
(c_i)_{i \in [n]} := (c_i^{(1)}, \sigma_i^{sym}, \text{Tag}_{mac}(\sigma_i^{sym})).
\]
\[ \text{Return } (c_i)_{i \in [n]} \]

Procedure `Dec(c^{(1)}, c^{(2)}, c^{(3)})`

\(\text{if } (c^{(1)})_{i \in [n]} = 1 \land \neg \text{calledEnc} \)
\[ \text{ABORTEarly} := \text{true} \]
\[ \text{else} \]
\[ \quad \text{if } \text{Enc}_{sym,mac}(c^{(2)}, c^{(3)}) = 1 \]
\[ \quad \quad \text{ABORTDec} := \text{true} \]
\[ \quad \text{else} \]
\[ \quad \quad \text{Return } \bot \]
\[ \quad \text{else} \]
\[ \quad \quad \text{if } \left(\exists (s, \cdot) \in L_H \text{ s.t. } \text{Enc}(s, c^{(1)}) = 1\right) \]
\[ \quad \quad \quad H(s) := (k_{sym, k_{mac}}) \]
\[ \quad \quad \quad \text{else} \]
\[ \quad \quad \quad \text{Add}(c^{(1)}, k_{sym, k_{mac}}) \text{ to } H_{patch} \]
\[ \quad \quad \quad \text{if } \text{Vrfy}_{mac}(c^{(2)}, c^{(3)}) = 1 \]
\[ \quad \quad \quad \text{Return } c_i^{(2)} \]
\[ \quad \quad \text{else} \]
\[ \quad \quad \text{Return } \bot \]

Procedure `Finalize(out)`

\[ \text{Return } \mathcal{R}((m_i)_{i \in [n]}, D, I, out) \]

---

**Fig. 6:** Reduction to KEM’s OW-PCA security given by the game interface for \(A\).

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

\(B\) runs \(A\) as \(A\) is run in game \(G_3\) except for the following differences: After calling `INITIALIZE`, \(B\) guesses an index \(i^{*} \overset{\$}{\leftarrow} [n]\); \(B\) creates \(c_i\) as before, but hides its own
challenge in the first component of the $i^{\text{th}}$ ciphertext. Let’s assume that $\text{ABORT}^H$ happens. Since $B$ knows $\{c_{i}^{(1)}\}$ for $i \in [n] \setminus \{i^*\}$, it can detect if $A$ queries $\text{HASH}(s)$ for $s \in \{k_i\}$ where $i \in [n] \setminus \{i^*\}$, while $B$ can invoke its $\text{CHECK}$ oracle to detect the query $\text{HASH}(k_{i^*})$ since $\text{CHECK}(k_{i^*}, c_{i}^{(1)}) = 1$. Therefore $B$ does not have to guess when $\text{ABORT}^H$ happens. If $A$ call $\text{OPEN}(i^*)$, $B$ apparently guessed $i^*$ wrong and aborts $A$. Running the reduction, $B$ has to maintain the conditions for $\text{ABORT}\text{DEC}$. Therefore it suffices to check if $c_{i}^{(1)} \in \{c_{i}^{(1)}\}_{i=1}^{n}$ and $\text{Vrfy}_{\text{sym}, \text{MAC}}(c^{(2)}, c^{(3)})$ hold, because $H(k)$ cannot be defined, since neither $\text{ABORT}^H$, nor $\text{ABORT}$ via $\text{OPEN}$ happened.

It remains to explain how $B$ (unable to compute $k = \text{Decap}_{sk}(c^{(1)})$) answers decryption queries without knowing $sk$. To answer these queries we make use of the nifty “oracle patching technique” from [18]. If $A$ calls $\text{DEC}(c^{(1)}, c^{(2)}, c^{(3)})$, $B$ 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 $s$, $B$ uses $(k_{\text{sym}}, k_{\text{MAC}}) := H(s)$. If not, $B$ picks $(k_{\text{sym}}, k_{\text{MAC}})$ at random and has to keep an eye on upcoming hash queries, since $B$ just committed to $H(k)$.

Therefore $B$ maintains a dedicated list $H_{\text{patch}}$ where $B$ adds $(c^{(1)}, (k_{\text{sym}}, k_{\text{MAC}}))$. On every hash query $\text{HASH}(s)$, $B$ checks if there is an entry $(c^{(1)}, k_{\text{sym}}, k_{\text{MAC}}) \in H_{\text{patch}}$ s.t. $\text{CHECK}(s, c^{(1)}) = 1$ in order to fix the oracle by setting $H(s) := (k_{\text{sym}}, k_{\text{MAC}})$. If $A$ should call $\text{DEC}(c^{(1)}_{i^*}, \ldots)$, challenger $C_3$ treats it like every other decryption query. Considering that $\text{ABORT}^H$ happens, $B$ only has to pick the right ciphertext to hide its own $\text{OW-PCA}$ challenge to win its game. Therefore $B$ succeeds if $\text{ABORT}^H$ happens and $B$ guessed $i^* \in [n]$ correctly:

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

Claim 5. There exists a simulator $S$ run in the $\text{IDEAL-SIM-SO-CCA}$ game such that $\text{Adv}(G^4, \text{IDEAL-SIM-SO-CCA}^S_{\text{PKE}, \text{MAC}}) = 0$.

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

Collecting the advantages of $A$ we get the claim as stated in Equation (1).
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 $G$ be a group of prime-order $p$, and let $g$ be a generator. The Diffie-Hellman KEM $DH$-KEM = (Gen, Enc, Dec) is defined as follows. The key-generation algorithm $\text{Gen}$ picks $x \xleftarrow{\$} \mathbb{Z}_p$ and defines $pk = X := g^x$ and $sk = x$; the encapsulation algorithm $\text{Encap}_{pk}$ picks $r \xleftarrow{\$} \mathbb{Z}_p$ and returns $(c = g^r, k = X^r)$; the decapsulation algorithm $\text{Decap}_{sk}(c)$ 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 $A$ that, given two random group elements $U := g^u, V := g^v$ and a restricted DDH oracle $O_v(a, b) := (a^v \overset{\$}{=} b)$ computes $g^{uv}$ with non-negligible probability.

Let $MAC$ be a MAC with message-space and key-space $\{0, 1\}^\ell$ and let $H : G \mapsto \{0, 1\}^{2\ell}$ be a family of hash functions. The security of $DHIES = PKE_{DH}$-KEM,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, $PKE_{RSA}$-KEM,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.

---

**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 $H : G \mapsto \{0, 1\}^{2\ell}$ be a family of hash functions. The security of $DHIES = PKE_{DH}$-KEM,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, $PKE_{RSA}$-KEM,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 $T = (GK, F, F^{-1})$, where $GK$ generates a key pair $(ek, td) \xleftarrow{\$} GK(1^\kappa)$, $F(ek, \cdot)$ implements a permutation

$$f_{ek} : \{0,1\}^k \rightarrow \{0,1\}^k$$

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Procedure</th>
<th>INITIALIZE($1^\kappa$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(ek, td) \xleftarrow{$} GK(1^\kappa)$</td>
<td>Return $ek$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Procedure</th>
<th>CHALLENGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(s, t) \xleftarrow{$} {0,1}^{\ell + k_1} \times {0,1}^{k_0}$</td>
<td>$y := F(ek, (s, t))$</td>
</tr>
<tr>
<td>Return $y$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Procedure</th>
<th>FINALIZE($s'$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return $(s \not\equiv s')$</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 8: PD-OW$_T$ game
4.2 Optimal Asymmetric Encryption Padding (OAEP)

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

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

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

  and outputs \( sk = td \) and \( pk = (ek, G, H) \).

- To encrypt a message \( m \in \{0, 1\}^\ell \), the sender draws a random value \( r \leftarrow \{0, 1\}^{k_0} \). Then it computes

  \[
  s = m || 0^{k_1} \oplus G(r) \quad t = r \oplus H(s).
  \]

  The ciphertext is \( C = F(ek, (s, t)) = f_{sk}(s, t) \).

- To decrypt a ciphertext \( C \), the decryption algorithm \( \text{Dec}_{sk}(c) \) 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.](image)

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 \( T \). 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 $T = (GK, F, F^{-1})$ 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 $R$, every adversary $A$ run in the REAL-SIM-SO-CCA$_{OAEP}$ 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 $S$ and an adversary $B$ in the PD-OW$_T$ experiment such that

$$Adv_{OAEP}^{SIM-SO-CCA}(A, S, R, n, \kappa) \leq \delta$$

where

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

Intuition for the proof of Theorem 9. We prove the theorem in a sequence of games, starting with the REAL-SIM-SO-CCA$_{OAEP}$ 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 IDEAL-SIM-SO-CCA$_{OAEP}$ 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 $\text{OPEN}(i)$. To this end, we have to describe how the challenger is able to create “non-committing” ciphertexts $c_1, \ldots, c_n$ in the ENC-procedure, which can then be opened to any message $m_i$ when $A$ issues an $\text{OPEN}(i)$-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 $A$. Here we use that $A$ 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 $A$.

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

Finally, we conclude with the observation that from $A$’s point of view all values of $H(s_i)$ remain equally likely until $\text{OPEN}(i)$ is asked, which implies also that it is very unlikely that $A$ ever asks $G(t_i \oplus H(s_i))$ before $\text{OPEN}(i)$. 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 $\text{OPEN}(i)$ is asked.

Proof of Theorem 9. The proof proceeds in a sequence of games, following [9, 43], where Game 0 corresponds to the REAL-SIM-SO-CCA$_{OAEP}^0$-experiment with adversary $A$ and a challenger, called $C_0$. From game to game, we gradually modify the challenger, until we obtain a challenger which is able to act as a simulator in the IDEAL-SIM-SO-CCA$_{OAEP}^S$ 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 \( q_{\text{int}} \) the number of decryption queries. For \( i \in [n] \) we will denote with \( c_i \) the \( i^{\text{th}} \) component of the challenge ciphertext vector \((c_i)_{i \in [n]}\), and we write \( e_i \) as \( e_i = f_{\text{sk}}(s_i, t_i) \).

<table>
<thead>
<tr>
<th>Procedure</th>
<th>INITIALIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>((ek, td) \xleftarrow{} GK(1^*))</td>
<td>Return ( ek )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Procedure</th>
<th>ENC(( \mathcal{D} ))</th>
</tr>
</thead>
</table>
| \((m_i)_{i \in [n]} \xleftarrow{} \mathcal{D}\) | for \( i \in [n] \):\[\begin{align*}
  r_i & \xleftarrow{} \{0, 1\}^{k_0} \\
  s_i & := m_i[0]^n \oplus G_{\text{int}}(r_i) \\
  t_i & := r_i \oplus H_{\text{int}}(s_i) \\
  c_i & := F(ek, (s_i, t_i))
\end{align*}\] Return \((c_i)_{i \in [n]}\) |

<table>
<thead>
<tr>
<th>Procedure</th>
<th>OPEN(i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I := I \cup {i})</td>
<td>Return ((m_i, r_i))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Internal procedure</th>
<th>(H_{\text{int}}(s))</th>
</tr>
</thead>
</table>
| If \((s, h_s) \notin L_H\) | \[\begin{align*}
  h_s & \xleftarrow{} \{0, 1\}^{k_0} \\
  L_H & := L_H \cup (s, h_s)
\end{align*}\] Return \( h_s \) |

<table>
<thead>
<tr>
<th>Internal procedure</th>
<th>(G_{\text{int}}(r))</th>
</tr>
</thead>
</table>
| if \((r, h_r) \notin L_G\) | \[\begin{align*}
  h_r & \xleftarrow{} \{0, 1\}^{k+1} \\
  L_G & := L_G \cup (r, h_r)
\end{align*}\] Return \( h_r \) |

<table>
<thead>
<tr>
<th>Procedure</th>
<th>(G(r))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(L_G^{\mathcal{A}} := L_G \cup {r})</td>
<td>Return ( G_{\text{int}}(r) )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Procedure</th>
<th>(\text{FINALIZE}(\text{out}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return ( \mathcal{R}((m_i)_{i \in [n]}, \mathcal{D}, I, \text{out}) )</td>
<td></td>
</tr>
</tbody>
</table>

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\[\begin{align*}
  L_G & \subseteq \{0, 1\}^{k_0} \times \{0, 1\}^{k+1} \\
  L_H & \subseteq \{0, 1\}^{k+1} \times \{0, 1\}^{k_0} \\
  L_{H}^{\mathcal{A}} & \subseteq \{0, 1\}^{k_0} \\
  L_{H}^{\mathcal{A}} & \subseteq \{0, 1\}^{k+1}
\end{align*}\]which are initialized to the empty set in the INITIALIZE procedure.

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

By definition we have\[\text{Adv}(\text{REAL-SIM-SO-CCA}_{\text{OAEP}}, G_0^{\mathcal{A}}) = 0.\]

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.
Game 1. In this game, $C_1$ proceeds exactly as $C_0$, except that instead of implementing procedure $\text{Dec}$, it uses procedure $\text{Dec}_1$ from Figure 11 to respond to decryption queries. Note that procedure $\text{Dec}_1$ does not require the trapdoor $td$ to perform decryption.

Claim 1. It holds that $\text{Adv}(G_0^A, G_1^A) \leq q_d \cdot \left(2^{-k_1} + q_g \cdot 2^{-k_0}\right)$.

Proof. Game 1 is perfectly indistinguishable from Game 0, unless $A$ makes a decryption query with ciphertext $c$, such that $\text{Dec}(c) \neq \text{Dec}_1(c)$. Note that this can only hold if $A$ 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 $A$’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 $A$’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 $\text{Adv}(G_0^A, G_1^A) \leq q_d \cdot \left(2^{-k_1} + q_g \cdot 2^{-k_0}\right)$.

Game 2. Challenger $C_2$ proceeds exactly like $C_1$, except that it implements procedure $\text{Enc}_2$ from Figure 11 instead of $\text{Enc}$. 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 $\text{Adv}(G_0^A, G_2^A) \leq n(q_g + n) \cdot 2^{-k_0}$.

Proof. Note that procedure $\text{Enc}_2$ first defines $r_i := H(s_i) \oplus t_i$ for uniformly random $t_i \leftarrow \{0, 1\}^{k_0}$. 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 \( \text{ENC}_2 \) does not terminate. In this case the hash function \( G \) is programmed such that \( G(r_i) = h_{r_i} = s_i \oplus m_i \| 0^{k_i} \). Since \( s_i \) is uniformly distributed, so is \( G(r_i) \), exactly as in Game 1. Thus, \( \text{ENC}_2 \) simulates procedure \( \text{ENC} \) 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 \( n(q_g + n) \cdot 2^{-k_0} \).

**Game 3.** We add an abort condition to the \( \text{OPEN} \)-procedure (see the left-hand side of Figure 12). Challenger \( C_3 \) proceeds exactly like \( C_2 \), except that it raises event \( \text{ABORT} \) and terminates, if \( A \) 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 \( \text{Adv}(G^A_2, G^A_3) \leq n \cdot q_h \cdot \text{Adv}_{pd}^T(B, \kappa) \).

**Proof.** Game 3 proceeds identically to Game 2, until event \( \text{ABORT} \) is raised. Thus we have

\[
\text{Adv}(G^A_2, G^A_3) \leq \text{Pr}[\text{ABORT}]
\]

We construct an adversary \( B \) against the partial-domain onewayness of \( T \). \( B \) receives as input \( ek \) and \( y = f_{ek}(s, t) \) for uniformly random \( (s, t) \overset{\$}{\leftarrow} \{0,1\}^{\ell+k_i} \times \{0,1\}^{k_0} \). It proceeds exactly like \( C_3 \), except for the following. At the beginning of the game it sets \( pk := ek \) and guesses two indices \( j \overset{\$}{\leftarrow} [n] \) and \( q \overset{\$}{\leftarrow} [q_h] \) uniformly random, and sets \( c_j := y \). Note that \( c_j \) is correctly distributed (cf. the changes introduced in Game 2).

When \( A \) makes its \( q^{th} \) query \( s^* \) to \( H \), then \( B \) returns \( s^* \) and terminates.

Assume that \( \text{ABORT} \) happens. Then, at some point in the game, \( A \) makes the first query \( s' \) to \( H \) such that \( s' = s_i \) is a partial-domain preimage of some \( c_j \). With probability \( 1/q_h \) it holds that \( s' = s_i \). Moreover, with probability \( 1/n \) we have \( i = j \).

In this case \( B \) obtains the partial preimage \( s = s_j \) of \( y = c_j \). Thus, \( B \) succeeds, if \( \text{ABORT} \) happens and if it has guessed \( j \in [n] \) and \( q \in [q_h] \) correctly. This happens with probability \( \text{Pr}[\text{ABORT}]/(n \cdot q_h) \), which implies that

\[
\text{Pr}[\text{ABORT}] \leq n \cdot q_h \cdot \text{Adv}_{pd}^T(B, \kappa).
\]
Game 4. We add another abort condition to the Finalize-procedure (see the right-hand side of Figure 12). Challenger \( C_4 \) raises event \texttt{ABORT\!R} and terminates, if \( A \) ever queries \( r_i \) to \( G_A \) for some \( i \in [n] \), before querying \texttt{OPEN}(i). Otherwise it proceeds like \( C_3 \).

Claim 4. It holds that \( \text{Adv}(G_3^A, G_4^A) \leq n \cdot q_y \cdot 2^{-k_1} \).

Proof. Note that \( A \) never queries \( s_i \) before querying \texttt{OPEN}(i) (or the game is aborted), due to the changes introduced in Game 3. Thus, for all \( i \not\in \mathcal{I} \), \( H(s_i) \) is uniformly random and independent of \( A \)'s view. Therefore, all \( r_i = t_i \oplus H(s_i) \) are uniformly random and independent of \( A \)'s view. Since \( A \) issues at most \( q_y \) queries to \( G \), and we have \( 1 \leq i \leq n \), this implies \( \text{Adv}(G_3^A, G_4^A) \leq n \cdot q_y \cdot 2^{-k_1} \).

Game 5. Note that the attacker in Game 4 never issues a query \( G(r_i) \) before asking \texttt{OPEN}(i), as otherwise the game is aborted. Thus, the challenger does not have to define the hash value \( G(r_i) \) before \texttt{OPEN}(i) is asked. Therefore we can move the definition of \( G(r_i) \) from the \texttt{ENC}_2\!-\!procedure to the \texttt{OPEN}\!-\!procedure.

Therefore we replace the procedures \texttt{ENC}_2 and \texttt{OPEN} from Game 4 with procedures \texttt{ENC} and \texttt{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 \texttt{ENC}\!-\!procedure, but in the \texttt{OPEN}\!-\!procedure. Moreover, this modification is completely oblivious to \( A \), which implies

\[
\text{Adv}(G_4^A, G_5^A) = 0.
\]

**Procedure \texttt{ENC}(\mathcal{D})**

\[
(m_i)_{i \in [n]} \leftarrow \mathcal{D}
\]

For \( i \in [n] \):

\[
s_i \xleftarrow{} \{0, 1\}^{\ell + k_1}, t_i \xleftarrow{} \{0, 1\}^{k_0}
\]

\[
c_i := F(ek, (s_i, t_i))
\]

\[
r_i := H(s_i) \oplus t_i
\]

if \( r_i \in L^A_G \)

\[
\text{ABORT}_G := \text{true}
\]

Return \((c_i)_{i \in [n]}\)

**Procedure \texttt{OPEN}(i)**

\[
\mathcal{I} := \mathcal{I} \cup \{i\}
\]

if \( s_i \in L^A_G \)

\[
\text{ABORT}_S := \text{true}
\]

if \( r_i \in L^A_G \)

\[
\text{ABORT}_R := \text{true}
\]

\[
h_{r_i} := s_i \oplus m_i \| 0^{k_1}
\]

\[
L_G := L_G \cup \{(r_i, h_{r_i})\}
\]

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 \texttt{OPEN}\!-\!procedure. This allows us to construct a simulator, whose procedures are described in Figure 14. Note that the view of \( A \) when interacting with the simulator is \textit{identical} to its view when interacting with challenger \( C_5 \), which implies

\[
\text{Adv}(G_5^A, G_6^A) = 0
\]
Fig. 14: Procedures used by the simulator to implement the REAL-SIM-SO-CCA\textsubscript{A\textsubscript{OAEP}} experiment. Instructions in boxes correspond to calls to the IDEAL-SIM-SO-CCA\textsubscript{A\textsubscript{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


