Chapter 7
Hybrid Signcryption

Tor E. Bjørstad

7.1 Background

A major limitation of many common asymmetric cryptographic primitives is that their computational efficiency is much worse than for corresponding symmetric-key algorithms. Hybrid cryptography is the branch of asymmetric cryptography that aims to overcome this weakness, by using symmetric primitives as components to improve the overall performance and flexibility of a larger asymmetric scheme.

The canonical example of the hybrid approach is hybrid encryption. In these schemes, a symmetric encryption algorithm, such as a block cipher in a secure mode of operation, is used to overcome the relative slowness and restricted message space of traditional public-key encryption schemes. Informally, this is done by using the public-key scheme to transmit a one-time symmetric key in a secure manner and using that key to encrypt subsequent communication with the symmetric cipher. This yields an overall scheme which is fast, efficient, and practical, even when encrypting long messages.

Although the basic concept of hybrid encryption has been common knowledge in the cryptographic community for many years, a formal construction paradigm was first suggested in the late 1990s by Cramer and Shoup [68]. Their KEM + DEM model splits a hybrid encryption scheme into two parts: an asymmetric *key encapsulation mechanism* (KEM) and a symmetric *data encapsulation mechanism* (DEM). The main benefit of this model is that the security of the KEM and DEM can be analyzed separately, under the knowledge that *generic* composition of a secure KEM and a secure DEM is essentially as secure as the component parts. Although not all hybrid encryption schemes fit into this framework, it has proven itself as a useful model for analysis in both theory and practice.

The original signcryption scheme proposed by Zheng [203] (as discussed in Sects. 3.3 and 4.3) is a natural example of the benefits of the hybrid approach in signcryption. Using a public-key signature scheme as his starting point, Zheng showed how to reap the benefits of both signatures and encryption at a low additional cost,

T. E. Bjørstad (✉)
Department of Informatics, The Selmer Center, University of Bergen, Bergen, Norway
e-mail: tor.bjorstad@ii.uib.no

A.W. Dent, Y. Zheng (eds.), *Practical Signcryption*, Information Security and Cryptography, DOI 10.1007/978-3-540-89411-7_7,

by using a symmetric key encryption scheme as a black-box component. A similar approach is used in many of the most efficient signcryption schemes in the literature (see Chaps. 4, 5, and 6). Hence it is of interest to study how hybrid techniques can be used to build signcryption schemes in a more general setting, to gain a better understanding of how these efficient schemes work.

It turns out that the formal analysis of hybrid signcryption schemes is more complicated than that of hybrid encryption. This stems from the increased complexity of obtaining message authenticity and integrity in addition to confidentiality. As discussed at depth in Chaps. 2 and 3, it is necessary to consider not only straightforward attacks against the authenticity and confidentiality of messages, but also more complex issues such as the distinction between outsider and insider attacks. As we shall see in Sects. 7.3 and 7.4 entirely different construction paradigms are needed to obtain appropriate models for outsider-secure and insider-secure hybrid signcryption.

A formal composition model for hybrid signcryption was first proposed by Dent in 2004, yielding an efficient model for signcryption KEMs in the outsider-secure setting [71, 73]. Dent's construction of outsider-secure signcryption KEMs is directly analogous to the corresponding construction of regular encryption KEMs. However, it is fundamentally impossible to produce an insider-secure signcryption KEM in this model. A model for insider-secure signcryption KEMs was also proposed by Dent in [71, 72]. This model covers Zheng's original scheme. However, this construction is quite complex and has a poor security reduction. This meant that the concrete security of Zheng's scheme appears significantly worse when analyzed in Dent's model, than in the non-hybrid setting of original security proof [12, 13, 36].

An improved model for insider-secure hybrid signcryption was given by Bjørstad and Dent [37], based on encryption tag-KEMs [5, 4] rather than regular encryption KEMs. As it turns out, this model provides a simpler description of signcryption schemes than its predecessor, and the generic security reduction for the signcryption tag-KEM + DEM construction is better. Zheng's signcryption scheme remains the canonical example of an (insider-secure) hybrid signcryption scheme, as it may be expressed in the signcryption tag-KEM + DEM setting with only a minor modification. In this model, the concrete security analysis of Zheng's scheme yields a similar result to that of the original proof of security [12, 13, 37].

The formal security analysis of hybrid signcryption has historically been performed in the simpler two-user (ADR) model presented in Chap. 2, rather than in a full multi-user (BSZ) setting presented in Chap. 3. Meanwhile, the multi-user security model is more suitable in the analysis of insider-secure signcryption schemes, where it is a reasonable assumption that an adversary may corrupt or otherwise obtain the private keys of legitimate users. A proof of security of signcryption tag-KEMs in the multi-user model may also have further applications, for example, in analysis of efficient key establishment protocols (discussed in Chap. 11). Initial study of the multi-user security of signcryption tag-KEMs was first performed by Yoshida and Fujiwara [200], although this text somewhat extends their results.

This chapter will commence by introducing the basic construction of hybrid encryption schemes in the KEM + DEM setting in Sect. 7.2. Following this, the

adaptation of hybrid encryption KEMs to outsider-secure signcryption KEMs will be described in Sect. 7.3. Finally, the use of tag-KEMs to describe insider-secure hybrid signcryption will be examined in detail in Sect. 7.4.

7.1.1 A Brief Word on Notation

This chapter contains many situations in which one algorithm (with access to one set of oracles) runs a second algorithm (with access to a different set of oracles) as a subroutine. In order for the main algorithm to simulate the correct execution environment for the sub-algorithm, the main algorithm must simulate the oracles to which the sub-algorithm is expecting access. This is rather cumbersome to write in the typical $\mathcal{A}^{\mathcal{O}}(x)$ notation; hence, we introduce a new notation for this chapter and write $\mathcal{A}(x; \mathcal{O})$ for an algorithm which takes as input x and has access to an oracle $\mathcal{O}$.

If we are writing out the definition of an algorithm $\mathcal{B}$ that runs an algorithm $\mathcal{A}(x; \mathcal{O})$ as a subroutine, we will first detail the algorithm $\mathcal{B}$ and then detail a second algorithm $\mathcal{O}$ which explains how $\mathcal{B}$ responds to $\mathcal{A}$'s oracle queries. In other words, $\mathcal{B}$ will run the sub-algorithm $\mathcal{A}$ and use the sub-algorithm $\mathcal{O}$ to respond to $\mathcal{A}$'s oracle queries. This allows for a more compact and easily readable presentation of the main algorithm.

7.2 Preliminaries

In order to study the construction of secure hybrid signcryption schemes, it is highly instructive to first consider the basic KEM + DEM framework used to model hybrid encryption schemes. As a part of this, the necessary properties of data encapsulation mechanisms (DEMs) used as black-box components in hybrid schemes are defined.

7.2.1 The Hybrid Framework

To serve as a gentle introduction to the world of hybrid cryptography, it is instructive to discuss briefly the traditional KEM + DEM framework for hybrid encryption schemes [68]. This framework nicely illustrates the basic methodology employed and will be built upon later when discussing more complex constructions used for signcryption. We begin by defining the basic building blocks.

Definition 7.1 (KEM) A key encapsulation mechanism $KEM = (\texttt{Setup}, \texttt{KeyGen}, \texttt{Encap}, \texttt{Decap})$ is a tuple of four algorithms:

- A probabilistic algorithm `Setup` that takes a security parameter 1^k as input, and returns some global information *param* that are common to all users of an instantiation of the scheme.
- A probabilistic algorithm `KeyGen` that takes the global information *param* as input and outputs a public/private keypair (sk, pk).

- A probabilistic algorithm Encap that takes a public key pk as input and outputs a pair (K, C), where K is a key and C is the encapsulation of K.
- A deterministic algorithm Decap that takes a private key sk and an encapsulation C as input and outputs either a key K or the unique error symbol $\perp$.

All variables may be represented as bitstrings of various lengths. In particular, the key K is a bitstring of a specific, fixed length determined by the security parameter. A KEM must be *sound*, in the sense that given a valid keypair (sk, pk) and a valid encapsulation $(K, C) \xleftarrow{R} \texttt{Encap}(pk)$, the output of $\texttt{Decap}(sk, C)$ will be K.

Definition 7.2 (DEM) A data encapsulation mechanism $DEM = (\texttt{Enc}, \texttt{Dec})$ is a tuple of two algorithms:

- A deterministic algorithm Enc that takes a key K and a message m as input and outputs a ciphertext C. We denote this $C \leftarrow \texttt{Enc}_K(m)$.
- A deterministic algorithm Dec that takes a key K and a ciphertext C as input and outputs either message m or the unique error symbol $\perp$. We denote this m or $\perp \leftarrow \texttt{Dec}_K(C)$.

The soundness criterion for a DEM is that the basic identity $m = \texttt{Dec}_K\big(\texttt{Enc}_K(m)\big)$ holds.

Given a KEM and DEM where the KEM outputs keys of suitable length for use with the DEM, a hybrid public-key encryption scheme (as defined in Sect. 1.3.3) can be constructed in a straightforward manner:

1. The Setup algorithm is run once to generate common information for all users.
2. Each user then runs KeyGen to generate their own public/private keypair.
3. When a sender S wants to transmit a message m to a receiver R, he computes $(K, C_1) \xleftarrow{R} \texttt{Encap}(pk_R)$ and encrypts the message as $C_2 \leftarrow \texttt{Enc}_K(m)$. The ciphertext $C \leftarrow (C_1, C_2)$ is then transmitted to R.
4. When the recipient R receives the ciphertext C from S, she extracts (C_1, C_2) from C, computes the symmetric key $K \leftarrow \texttt{Decap}(sk_R, C_1)$, and obtains the message $m \leftarrow \texttt{Dec}_K(C_2)$.

The above construction is a sound encryption scheme assuming the soundness of the KEM and DEM. Our main benefit of separating the encryption scheme into a KEM and a DEM is that the security of the components can be analyzed separately. Considering the basic building blocks instead of the entire scheme simplifies analysis and allows hybrid encryption schemes to be tailor-made, since choice of KEM and DEM can be made independently.

In order to build signcryption schemes instead of encryption schemes, it is tempting to start by modifying the basic specification of a KEM given in Definition 7.1 and changing as little as possible. As we will observe in Sect. 7.3.1 this leads us to Dent's basic framework for outsider-secure signcryption KEMs [71, 73]. Before we look at this, however, it is necessary to define what sort of security criteria we need the DEMs to fulfill in order to use them in building hybrid signcryption schemes.

7.2.2 Security Criteria for Data Encapsulation Mechanisms

Whereas the main goal of Sects. 7.3 and 7.4 will be to work out secure alternatives to KEMs that can be used to build efficient signcryption schemes, the DEM of Definition 7.2 shall largely be left alone. This has a perfectly reasonable explanation: When attempting to create a new type of public-key scheme based on models for hybrid encryption, our goal is best reached by altering the public-key component used. However, before we can start discussing hybrid signcryption schemes, we must first define which security properties we expect a secure DEM to fulfill.

As our requirements will differ in the case of insider-secure and outsider-secure signcryption, several different requirements will be given. In practice, all these requirements may be realized by a secure symmetric encryption algorithm (such as AES-CTR), possibly together with an authentication mechanism in form of a message authentication code (MAC) [68]. As in the case of regular (non-hybrid) signcryption, we distinguish between security criteria required for a scheme to provide confidentiality and criteria required to provide authenticity and integrity.

The standard notion of confidentiality in cryptography is that of indistinguishability (IND). In the particular case of data encapsulation mechanisms, the two notions that we are interested in are those of one-time IND-CPA security and one-time IND-CCA security, described in Sect. 1.3.4. As we shall see, almost paradoxically, we will require IND-CCA-secure DEMs for constructing outsider-secure signcryption schemes and IND-CPA-secure DEMs for constructing insider-secure signcryption schemes.

With respect to authenticity and integrity, we define a DEM to be integrally secure (INT-CCA) if there is no efficient adversary that can create valid ciphertexts C. This corresponds to the usual notion of unforgeability and gives a receiver faith that a valid ciphertext must have been generated legitimately. In practice, this is usually achieved by using a MAC. As we will see, INT-CCA security is only required for the outsider-secure hybrid constructions. The INT-CCA game between the challenger and adversary $\mathcal{A}$ is quite simple and runs as follows.

1. The challenger generates a random symmetric key K^* of appropriate length for the security parameter.
2. The adversary runs $\mathcal{A}$ on the input 1^k. When $\mathcal{A}$ terminates, it outputs a ciphertext C^*. During its execution, $\mathcal{A}$ may query an encryption oracle that for a given input message m outputs $\mathtt{Enc}_{K^*}(m)$ and a decryption oracle that for a given ciphertext C outputs $\mathtt{Dec}_{K^*}(C)$.

The adversary wins the game whenever $\mathtt{Dec}_K(C^*) \neq \perp$ and C^* was never output by the encryption oracle. The advantage of $\mathcal{A}$ is simply $Pr[\mathcal{A}\text{ wins}]$.

Definition 7.3 (Unforgeable DEM) We say that a DEM is unforgeable (INT-CCA secure) if the advantage of any polynomial-time adversary in the INT-CCA game is negligible with respect to the security parameter k.

7.3 Hybrid Signcryption with Outsider Security

Signcryption schemes with outsider security are useful for communication between a set of trusted parties, as they are both more efficient with respect to computational cost and simpler to design and analyze than their insider-secure counterparts. The problem of constructing a framework for outsider-secure hybrid signcryption was first considered by Dent [71, 73] and can be solved by a fairly straightforward adaptation of the KEM/DEM construction for hybrid encryption discussed in Sect. 7.2.1. Although outsider-secure signcryption has largely been overlooked in the research literature, we believe that these schemes are useful and have practical applications. Our treatment closely follows Dent's original.

7.3.1 An Outsider-Secure Signcryption KEM

The main idea behind Dent's outsider-secure signcryption KEM [71, 73] is to use the traditional encryption KEM (described in Sect. 7.2.1) as a starting point and alter as little as possible to obtain something that behaves like as a signcryption scheme. This is reasonably straightforward: Instead of a single algorithm `KeyGen`, we should specify two algorithms, one used to generate sending ("signing") keys and a separate algorithm to generate keys for receiving ("decrypting") messages. Furthermore, the encapsulation algorithm must now take the private key of the sender *and* the public key of the receiver as input, and vice versa for the decapsulation algorithm. This leads directly to the following specification of an outsider-secure signcryption KEM (*SKEM*).

Definition 7.4 (**Signcryption KEM**) An (outsider-secure) signcryption KEM $SKEM = (\texttt{Setup}, \texttt{KeyGen}_S, \texttt{KeyGen}_R, \texttt{Encap}, \texttt{Decap})$ is a tuple of five algorithms.

- A probabilistic algorithm `Setup` that takes a security parameter 1^k as input and returns some global information *param* that are common to all users of an instantiation of the scheme.
- A probabilistic algorithm $\texttt{KeyGen}_S$ that takes the global information *param* as input and outputs a public/private keypair (sk_S, pk_S) used for sending messages.
- A probabilistic algorithm $\texttt{KeyGen}_R$ that takes the global information *param* as input and outputs a public/private keypair (sk_R, pk_R) used for receiving messages.
- A probabilistic algorithm `Encap` that takes the sender's private key sk_S and the receiver's public key pk_R as input, and outputs a pair (K, C), where K is a key and C is the encapsulation of K.
- A deterministic algorithm `Decap` that takes the sender's public key pk_S, the receiver's private key sk_R, and a key encapsulation C as input, and outputs either a symmetric key K or the unique error symbol $\perp$.

By combining the signcryption KEM with a standard DEM, we obtain a hybrid signcryption scheme in the obvious manner.

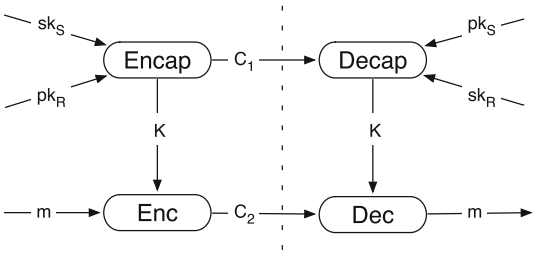

Fig. 7.1 Data flow in the outsider-secure signcryption KEM + DEM construction

Definition 7.5 (**SKEM+DEM hybrid signcryption scheme**) Suppose that (`Setup`, $\texttt{KeyGen}_S$, $\texttt{KeyGen}_R$, `Encap`, `Decap`) is a signcryption KEM and (`Enc`, `Dec`) is a DEM and that the keys produced by the signcryption KEM are of appropriate length for use with the DEM for all security parameters k. Then we can construct a hybrid signcryption scheme by using the `Setup`, $\texttt{KeyGen}_S$, and $\texttt{KeyGen}_R$ algorithms from the SKEM and defining the algorithms `Signcrypt` and `Unsigncrypt` as follows:

- The `Signcrypt` algorithm takes as input the private key of the sender sk_S, the public key of the receiver pk_R, and a message m. It computes $(K, C_1) \xleftarrow{R} \texttt{Encap}(sk_S, pk_R)$ and $C_2 \leftarrow \texttt{Enc}_K(m)$ and outputs the signcryptext $C \leftarrow (C_1, C_2)$.
- The `Unsigncrypt` algorithm takes as input the public key of the sender pk_S, the private key of the receiver sk_R, and a signcryptext C. It parses C to obtain (C_1, C_2) and computes $\texttt{Decap}(pk_S, sk_R, C_1)$. If `Decap` returned $\perp$, then the algorithm must output $\perp$ and halt. Otherwise, it computes $\texttt{Dec}_K(C_2)$. The output of `Dec` is either $\perp$ or a message m, in either case the algorithm outputs the result and halts.

The data flow between the `Signcrypt` and `Unsigncrypt` algorithms is illustrated in Fig. 7.1.

7.3.2 Security Criteria for Outsider-Secure Signcryption KEMs

The main advantage of the KEM + DEM construction paradigm is that we may analyze the security of the KEM and DEM separately, with no significant loss of concrete security. It is therefore necessary to give a precise specification of what it means for a signcryption KEM to be secure. To attain outsider security we require that the SKEM preserves the confidentiality of encapsulated keys, which is the same as the confidentiality requirement for encryption KEMs [68]. Additionally, a signcryption KEM must preserve the authenticity and integrity of the encapsulated key, to ensure that some third party may not alter the encapsulated key in any meaningful fashion. These security notions are expressed in the usual manner, by way of formal attack games.

With respect to confidentiality, we adapt the indistinguishability criterion to the signcryption KEM setting. More precisely, we want that any polynomial-time adversary $\mathcal{A}$ is unable to distinguish between a real key K_0 output by the Encap algorithm from a key K_1 drawn uniformly at random from the set of possible keys. For a given security parameter k, this may be expressed through the following game between the challenger and a two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

1. The challenger runs the appropriate algorithms to generate some global information *param* and private/public keys for the sender and the receiver, denoted (sk_S, pk_S) and (sk_R, pk_R), respectively.
2. The adversary runs $\mathcal{A}_1$ on the input $(param, pk_S, pk_R)$. During its execution, $\mathcal{A}_1$ may query two oracles:
   - The encapsulation oracle $\mathcal{O}_{Encap}$ takes an arbitrary public receiving key pk as input and returns the result of computing $\texttt{Encap}(sk_S, pk)$.
   - The decapsulation oracle $\mathcal{O}_{Decap}$ takes an arbitrary public sending key pk and an encapsulation C as input and returns the result of computing $\texttt{Decap}(pk, sk_R, C)$.

   The algorithm terminates by outputting some state information *state*.
3. The challenger generates a valid encapsulation $(K_0, C^*) \xleftarrow{R} \texttt{Encap}(pk_S, sk_R)$, as well as a random key K_1 of the correct length. It then chooses a random bit $b \xleftarrow{R} \{0, 1\}$ and fixes the challenge encapsulation as (K_b, C^*).
4. The adversary runs $\mathcal{A}_2$ on the input $(K_b, C^*, state)$. During its execution, $\mathcal{A}_2$ may query the same oracles as before, with the restriction that it may not query the decapsulation oracle on the challenge encapsulation (pk_S, C^*). It terminates by returning a guess b' for the value of b.

The adversary wins the game if $b = b'$. The adversary's advantage is defined to be $|Pr[b = b'] - 1/2|$.

Definition 7.6 (**Indistinguishable signcryption KEM**) A signcryption KEM is said to be indistinguishable (IND-CCA2 secure) if the advantage of any polynomial-time adversary $\mathcal{A}$ in the IND-CCA2 game is negligible with respect to the security parameter k.

With respect to authenticity and integrity, Dent defines the security criterion in terms of indistinguishability of the real signcryption KEM and an *ideal* version of the same [71, 73]. This definition may seem somewhat unusual, as it is more common to see authenticity criteria specified in terms of an unforgeability requirement. However, an adversary creating forgeries of a signcryption KEM may in fact be used to distinguish said SKEM from an ideal one [71, 73]. As we will see, the definition using the notion of an ideal signcryption KEM turns out to be precisely what is needed to prove that our hybrid signcryption in Definition 7.5 is outsider secure. It also makes a nice parallel to the previous definition of IND-CCA2 security. For confidentiality, we needed the keys output by the encapsulation algorithm to be indistinguishable from random keys, the requirement for authenticity and integrity is that the entire signcryption KEM is indistinguishable from a random (i.e., ideal) one.

Given a signcryption KEM $SKEM = (\texttt{Setup}, \texttt{KeyGen}_S, \texttt{KeyGen}_R, \texttt{Encap}, \texttt{Decap})$ we define the corresponding ideal signcryption KEM to be the five-tuple of algorithms $Sim.SKEM = (\texttt{Sim.Setup}, \texttt{KeyGen}_S, \texttt{KeyGen}_R, \texttt{Sim.Encap}, \texttt{Sim.Decap})$, together with an internal state list *KeyList* containing key/encapsulation pairs. The simulated algorithms are defined as follows:

- The simulated setup algorithm `Sim.Setup` takes the security parameter 1^k as input and runs `Setup` to obtain the global information *param*. It then initializes *KeyList* as an empty list and returns *param*.
- The simulated encapsulation algorithm `Sim.Encap` takes the keys sk_S and pk_R as input. It then performs the following steps:

  1. Compute an encapsulation (K, C) using the real encapsulation algorithm $\texttt{Encap}(sk_S, pk_R)$.
  2. Check whether there exists a pair (K', C) in *KeyList*. If this is the case, the algorithm returns K' and halts.
  3. Otherwise, the algorithm generates a new K' of appropriate length uniformly at random, adds (K', C) to *KeyList*, returns K', and halts.

- The simulated decapsulation algorithm `Sim.Decap` takes the keys pk_S and sk_R together with an encapsulation C as input. It then performs the following steps:

  1. Check whether there exists a pair (K, C) in *KeyList*. If this is the case, the algorithm returns K and halts.
  2. Otherwise, the algorithm runs the real decapsulation algorithm `Decap` (pk_S, sk_R, C). If the decapsulation fails and outputs $\perp$, the algorithm returns $\perp$ and halts.
  3. If `Decap` did not return $\perp$, the algorithm generates a new K of appropriate length uniformly at random, adds (K, C) to *KeyList*, returns K, and halts.

It is clear from the above specification that the simulated signcryption KEM is self-consistent. Furthermore, it is "ideal" in the sense that we desire: an encapsulation C reveals *no* information about the encapsulated key K (since the key is chosen uniformly at random and independently of C). We say that the signcryption KEM is *left-or-right* secure (LoR-CCA) if there is no efficient algorithm to distinguish between the real and the idealized signcryption KEMs. For a given security parameter k, the LoR-CCA game proceeds as follows:

1. The challenger picks a bit $b \xleftarrow{R} \{0, 1\}$ at random.
2. The challenger generates global information *param*, either by running `Setup` if b was 0 or by running `Sim.Setup` if b was 1. The challenger then generates private/public keys for the sender and the receiver in the ordinary manner using $\texttt{KeyGen}_S$ and $\texttt{KeyGen}_R$.
3. The adversary runs $\mathcal{A}$ on the input (pk_S, pk_R). During its execution, $\mathcal{A}$ may query decapsulation and encapsulation oracles as specified in the previous IND-CCA2 game. However, the responses to $\mathcal{A}$'s queries are computed using the *real*

`Encap` and `Decap` algorithms if $b = 0$ and the *ideal* algorithms `Sim.Encap` and `Sim.Decap` if $b = 1$. $\mathcal{A}$ terminates by outputting a guess b' for the value of b.

The adversary wins the game if $b = b'$. The adversary's advantage is defined as $|\Pr[b = b'] - 1/2|$.

Definition 7.7 (**LoR-CCA-secure signcryption KEM**) A signcryption KEM is said to be left-or-right (LoR-CCA) secure if the advantage of any polynomial-time adversary $\mathcal{A}$ in the LoR-CCA game is negligible with respect to the security parameter k.

Definition 7.8 (**Outsider-secure signcryption KEM**) A signcryption KEM is said to be *outsider secure* if it is both indistinguishable and left-or-right secure.

7.3.3 Security of the SKEM + DEM Construction

Having specified the security models in use for a signcryption KEM and DEM, it remains to show that the hybrid signcryption scheme of Definition 7.5 is an outsider-secure signcryption scheme satisfying the relevant security criteria defined in Chap. 3. The proof of this is quite straightforward and is quite similar to the original proof that hybrid *encryption* schemes are IND-CCA2 secure given in [68]. Since both security models for the signcryption KEMs are based on indistinguishability of certain attributes of the KEM from random in the view of the outside attacker, we state a well-known lemma used in the proofs. This can be thought of as a more general version of Lemma 1.1.

Lemma 7.1 *(Distinguisher lemma) Let G_0 and G_1 be two games. Suppose that an experimenter picks $b \xleftarrow{R} \{0, 1\}$ uniformly at random and proceeds to play G_b with a distinguisher algorithm that outputs a guess b' of the value of b. Then*

$$2\left|Pr[b = b'] - 1/2\right| = \left|Pr[b' = 0|b = 0] - Pr[b' = 0|b = 1]\right|. \tag{7.1}$$

The result follows from simple manipulation of conditional probabilities, see, for example, [71]. We proceed to prove that Dent's outsider-secure signcryption KEM can be used to build an outsider-secure hybrid signcryption scheme.

Theorem 7.1 *(Security of SKEM + DEM hybrid signcryption) Let SC be a hybrid signcryption scheme constructed from a signcryption KEM (Definition 7.4) and a DEM (Definition 7.2). If the signcryption KEM is IND-CCA2 secure and the signcryption DEM is one-time IND-CCA secure, then the hybrid signcryption scheme is multi-user outsider FSO/FUO-IND-CCA2 secure (Definition 3.1) with the bound*

$$\varepsilon_{SC,IND\text{-}CCA2} \leq 2\,\varepsilon_{SKEM,IND\text{-}CCA2} + \varepsilon_{DEM,IND\text{-}CCA} \tag{7.2}$$

where the ε values denote the maximal success probability of adversaries in the specified attack games. Furthermore, if the signcryption KEM is LoR-CCA secure

and the signcryption DEM is INT-CCA secure, then the hybrid signcryption scheme is multi-user outsider FSO/FUO-sUF-CMA secure (Definition 3.2) with the bound

$$\varepsilon_{SC,sUF\text{-}CMA} \leq 2\,\varepsilon_{SKEM,LoR\text{-}CCA} + \varepsilon_{DEM,INT\text{-}CCA} \tag{7.3}$$

Proof The proofs of the two statements are remarkably similar. In both cases, we proceed by modifying the original (FSO/FUO-IND-CCA2 or FSO/FUO-sUF-CMA) attack game for SC in a way that relates to the corresponding (IND-CCA2 or LoR-CCA) security criterion for the signcryption KEM. Lemma 7.1 is used to do this. Finally, we show that the adversary must break the (IND-CCA or INT-CCA) security of the DEM to gain any advantage in the modified game. We consider first the case of indistinguishability.

Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary against the FSO/FUO-IND-CCA2 security of SC, G_0 be the regular FSO/FUO-IND-CCA2 game for outsider-secure signcryption as given by Definition 3.1, and X_0 the event that the adversary wins in G_0. Next we define a modified game G_1. The difference between G_0 and G_1 is that the challenge ciphertext is computed using a *random* symmetric key K_1. In other words, the challenge ciphertext $C^* = (C_1^*, C_2^*)$ is constructed by computing $(K_0, C_1^*) \xleftarrow{R} \texttt{Encap}(sk_S, pk_R)$, drawing another key K_1 uniformly at random from the keyspace, and then using it to compute $C_2^* \leftarrow \texttt{Enc}_{K_1}(m)$. In order to remain consistent, the challenger should also use K_1 to answer any unsigncryption oracle query of the form $\big(pk_S, (C_1^*, \cdot)\big)$. Hence, the difference between G_0 and G_1 lies solely in how the signcryption KEM operates. The two games correspond to the situations $b = 0$ and 1 in the IND-CCA2 game against the signcryption KEM.

Let X_1 be the event that $\mathcal{A}$ wins G_1. We argue that probability $|\Pr[X_0] - \Pr[X_1]|$ is bounded by $2\varepsilon_{SKEM,IND\text{-}CCA2}$, where $\varepsilon_{SKEM,IND\text{-}CCA2}$ is the advantage of a specific adversary $\mathcal{D}$ against the IND-CCA2 security of the signcryption KEM used to construct SC. The idea is that the distinguisher $\mathcal{D}$ plays either G_0 or G_1 with a regular adversary $\mathcal{A}$ against the full signcryption scheme, depending on the value of the hidden bit b which $\mathcal{D}$ is trying to determine. By Lemma 7.1, any non-negligible difference in the advantage of $\mathcal{A}$ can be leveraged by $\mathcal{D}$ to break the signcryption KEM, and the stated bound is obtained for the game transition.

Next, consider the probability that X_1 does in fact occur. We argue that this is the same as $\varepsilon_{DEM,IND\text{-}CCA}$. This follows from the way that G_1 is defined. The first part of the challenge C_1^* reveals no direct information about which message was signcrypted, since the symmetric key K_1 was chosen independently and uniformly at random. Thus, to gain a non-negligible advantage in G_2, the adversary must somehow learn something from the symmetric ciphertext C_2^*. The adversary is able to mount a chosen ciphertext attack, since decryption oracle queries of the form $\big(pk_S, (C_1^*, \cdot)\big)$ must be decrypted using K_1 to maintain consistency. More formally, we show by construction that an adversary $\mathcal{A}$ playing the game G_1 can be converted into an IND-CCA adversary $\mathcal{B}$ against the DEM with essentially the same advantage. A specification of such an adversary is shown in Fig. 7.3.

To summarize, we have shown that the difference $|\Pr[X_0] - \Pr[X_1]|$ is bounded by $2\varepsilon_{SKEM,IND\text{-}CCA2}$ while $Pr[X_1]$ itself is essentially equal to $\varepsilon_{DEM,IND\text{-}CCA}$, thus obtaining the stated bound.

For authenticity and integrity, the proof is highly similar in both approach and execution and will therefore not be specified in the same level of detail. Again we consider an adversary $\mathcal{A}$, this time attacking the FSO/FUO-sUF-CMA security of SC. Again, we let G_0 to be the regular FSO/FUO-sUF-CMA attack game given by Definition 3.2 and X_0 to be the event that $\mathcal{A}$ wins G_0. Our subsequent game G_1 is similar to G_0, but modified so that an *ideal* signcryption KEM is used instead of the regular one. It is straightforward to construct a new distinguisher similar to the one given in Fig. 7.2, which relates the difference between G_0 and G_1 to the advantage of a LoR-CCA adversary against the signcryption KEM. Furthermore, the advantage

$\mathcal{D}_1(param, pk_S, pk_R; \mathcal{O}_{Encap}, \mathcal{O}_{Decap})$:
$(m_0, m_1, s) \xleftarrow{R} \mathcal{A}_1(param, pk_S, pk_R; \mathcal{O}_{SC}, \mathcal{O}_{USC})$.
$state \leftarrow (m_0, m_1, s)$.
Return state.

$\mathcal{D}_2(K^*, C_1^*, \text{state}; \mathcal{O}_{Encap}, \mathcal{O}_{Decap})$:
Parse $state$ as (m_0, m_1, s).
$b \xleftarrow{R} \{0,1\}$.
$C_2^* \leftarrow \mathtt{Enc}_{K^*}(m_b)$.
$C^* \leftarrow (C_1^*, C_2^*)$.
$b' \xleftarrow{R} \mathcal{A}_2(C^*, s; \mathcal{O}_{SC}, \mathcal{O}_{USC})$.
If $b = b'$ then return 1.
Else return 0.

$\mathcal{O}_{SC}(pk, m)$:
$(K, C_1) \xleftarrow{R} \mathcal{O}_{Encap}(pk)$.
$C_2 \leftarrow \mathtt{Enc}_K(m)$.
$C \leftarrow (C_1, C_2)$.
Return C.

$\mathcal{O}_{USC}(pk, C)$:
$(C_1, C_2) \leftarrow C$.
If $pk = pk_R$ and $C_1 = C_1^*$ then
  $K \leftarrow K^*$.
Else if $\perp = \mathcal{O}_{Decap}(pk, C_1)$ then
  Return $\perp$ and halt.
Else $K \leftarrow \mathcal{O}_{Decap}(pk, C_1)$.
If $\perp = \mathtt{Dec}_K(C_2)$ then
  Return $\perp$ and halt.
Else $m \leftarrow \mathtt{Dec}_K(C_2)$.
Return m.

Fig. 7.2 A complete specification of the distinguisher algorithm $\mathcal{D}$ for the SKEM

$\mathcal{B}_1(1^k; \mathcal{O}_{Dec})$:
$param \xleftarrow{R} \mathtt{Setup}(1^k)$.
$(sk_S, pk_S) \xleftarrow{R} \mathtt{KeyGen}_S(param)$.
$(sk_R, pk_R) \xleftarrow{R} \mathtt{KeyGen}_R(param)$.
$(m_0, m_1, s) \xleftarrow{R} \mathcal{A}(param, pk_S, pk_R; \mathcal{O}_{SC}, \mathcal{O}_{USC})$.
$state \leftarrow (param, sk_S, pk_S, sk_R, pk_R, m_0, m_1, s)$.
Return $(m_0, m_1, state)$.

$\mathcal{B}_2(C_2^*, state; \mathcal{O}_{Dec})$:
Parse $state$ as
  $(param, sk_S, pk_S, sk_R, pk_R, m_0, m_1, s)$.
$(K, C_1^*) \xleftarrow{R} \mathtt{Encap}(sk_S, pk_R)$.
$C^* \leftarrow (C_1^*, C_2^*)$.
$b \xleftarrow{R} \mathcal{A}_2(C^*, s; \mathcal{O}_{SC}, \mathcal{O}_{USC})$.
Return b.

$\mathcal{O}_{SC}(pk, m)$:
$(K, C_1) \xleftarrow{R} \mathtt{Encap}(sk_S, pk)$.
$C_2 \leftarrow \mathtt{Enc}_K(m)$.
$C \leftarrow (C_1, C_2)$.

$\mathcal{O}_{USC}(pk, C)$:
$(C_1, C_2) \leftarrow C$.
If $C_1 = C_1^*$ and $pk = pk_R$ then
  Return $\mathcal{O}_{Dec}(C_1)$.
Else if $\perp = \mathtt{Decap}(pk, sk_R, C_1)$ then
  Return $\perp$ and halt.
Else $K \leftarrow \mathtt{Decap}(pk, sk_R, C_1)$.
If $\perp = \mathtt{Dec}_K(C_2)$
  Return $\perp$ and halt.
Else $m \leftarrow \mathtt{Dec}_K(C_2)$.
Return m.

Fig. 7.3 A complete specification of the distinguisher algorithm $\mathcal{B}$ for the DEM

of $\mathcal{A}$ in G_1 can be shown to be bounded by that of an INT-CCA adversary against the DEM, by a construction similar to that of Fig. 7.3. This concludes the proof. □

Although we have established that the combination of a signcryption KEM and DEM can be used to build outsider-secure hybrid signcryption schemes, more complex constructions are needed for insider security. This stems from the observation that there is no connection between the encapsulations generated by `Encap` and the actual message that is being signcrypted. In fact, the receiver can create a valid signcryptext for an arbitrary message m given a single valid signcryption $C = (C_1, C_2)$, by computing $K \leftarrow \texttt{Decap}(pk_S, sk_R, C_1)$ and computing a new value $C_2' \leftarrow \texttt{Enc}_K(m)$. Hence the scheme is trivially forgeable by an inside attacker and has no way of providing non-repudiation. From this, we observe that in order to build insider-secure hybrid signcryption the signcryption KEM must somehow prevent the adversary from tampering with m or C_2.

7.3.4 Outsider-Secure Hybrid Signcryption in Practice

Outsider-secure signcryption has not been the target of much research since the distinction was first recognized by An et al. [10]. This is unfortunate, as it is possible to construct outsider-secure schemes that are simpler and more efficient than their insider-secure counterparts. These schemes would clearly be suitable for any real-world settings where insider attacks are not part of the threat model. The only known outsider-secure signcryption KEM was proposed by Dent in [71, 73]. It is extremely simple, has a low additional computational cost, and is based on the well-known ECIES encryption KEM [2, 101]. The ECISS[1]-KEM is specified in Fig. 7.4. In the two-user setting, this scheme has been proven to be secure (in the random oracle model), with respect to the computational Diffie–Hellman problem in the underlying group[2] [71, 73]. A tighter bound can be obtained by considering the security relative to the Gap Diffie–Hellman problem instead.

$\texttt{KeyGen}_S(param)$:
$sk_S \xleftarrow{R} \mathbb{Z}_q^*$.
$pk_S \leftarrow g^{sk_S}$.
Return (sk_S, pk_S).

$\texttt{KeyGen}_R(param)$:
$sk_R \xleftarrow{R} \mathbb{Z}_q^*$.
$pk_R \leftarrow g^{sk_R}$.
Return (sk_R, pk_R).

$\texttt{Encap}(sk_S, pk_R)$:
$r \xleftarrow{R} \mathbb{Z}_q^*$.
$C \leftarrow g^r$.
$\kappa \leftarrow pk_R^{sk_S} \cdot g^r$.
$K \leftarrow H(\kappa, pk_S, pk_R)$.
Return (K, C).

$\texttt{Decap}(pk_S, sk_R, C)$:
$\kappa \leftarrow pk_S^{sk_R} \cdot C$.
$K \leftarrow H(\kappa, pk_S, pk_R)$.
Return K.

Fig. 7.4 A complete specification of the ECISS-KEM

[1] ECISS stands for elliptic-curve integrated signcryption scheme.
[2] Note that the alternate scheme suggested without a security proof in [71] is insecure [92].

The ECISS scheme is a good example of how outsider-secure signcryption can be obtained at low additional cost compared to regular encryption and underlines the close relationship between outsider-secure signcryption KEMs and secure encryption KEMs. Comparing ECISS-KEM and ECIES-KEM, the only significant difference lies in how the input to the key derivation function H is computed. In the encryption-only scheme, the shared value is computed from the (receiver's) public key and the random value r as $pk^r = g^{sk \cdot r}$. For signcryption, the Diffie–Hellman value $pk_R^{sk_S} = pk_S^{sk_R} = g^{sk_S \cdot sk_R}$ is multiplied with g^r instead.

The original proof that ECISS-KEM is secure in the two-user model from [71] may readily be extended to the multi-user setting. However, to keep the reduction tight it is necessary to make the proof relative to the Gap Diffie–Hellman problem. One minor change to the original scheme is also required, namely that the public keys of the sender and receiver are included as input to the hash function. This has little practical significance, but enables us to keep sessions between different pairs of users distinct in the proof. As the proofs for IND-CCA2 and LoR-CCA security are almost identical, only the former will be shown here.

Theorem 7.2 *(Multi-user security of ECISS) The ECISS signcryption KEM is IND-CCA2 secure in the random oracle model, with respect to the Gap Diffie–Hellman problem. In particular, let $\mathcal{A}$ be an adversary that breaks the IND-CCA2 security of ECISS-KEM with advantage ε_{KEM}, while making at most q_E encapsulation and q_D decapsulation oracle queries. Then there exists an algorithm $\mathcal{B}$ solving the GDH problem whose advantage is given by*

$$\varepsilon_{\text{KEM}} \leq \varepsilon_{\text{GDH}} + \frac{q_E + q_D}{q}. \tag{7.4}$$

Proof Let $\mathcal{B}$ be an algorithm which tries to solve the Gap Diffie–Hellman problem (as defined in Sect. 4.1) in $\mathbb{G}$. The algorithm receives as input two random group elements g^a and g^b and will try to compute g^{ab} by using an adversary $\mathcal{A}$ against the ECISS signcryption KEM as a subroutine. During its execution, $\mathcal{B}$ may query a DDH oracle on triplets (g^x, g^y, g^z) which tests whether $g^{xy} = g^z$.

Our approach will be to use the challenge values g^a and g^b in place of the public keys pk_S and pk_R. This means that sk_S and sk_R will not be known to $\mathcal{B}$ and thus we have to be careful when simulating the encapsulation and decapsulation oracles. Partial consistency is maintained through our simulation of the key derivation function H as a random oracle and using the DDH oracle to verify that the correct relation between the public keys, C and κ, are maintained. The goal of $\mathcal{B}$ is to obtain values C and κ such that $C \cdot g^{ab} = \kappa$, in which case g^{ab} can be recovered.

We will use two lists to keep track of oracle queries by $\mathcal{A}$. As opposed to [71], it will also be necessary to keep track of the public keys used in the oracle queries. Let *EncapList* be a list of tuples (pk_S, pk_R, C, K) and *HashList* be a list of tuples (pk, pk', κ, K). It is necessary to specify how $\mathcal{B}$ should respond to queries from $\mathcal{A}$ to the encapsulation, decapsulation, and random oracles, so that these responses are self-consistent and follow the correct distributions:

- For an encapsulation oracle query pk, $\mathcal{B}$ should first pick a random group element C. If there is an entry (pk_S, pk, C, K) in *EncapList*, then output the pair (C, K). Otherwise, if there is an entry (pk_S, pk, κ, K) in *HashList* such that $(pk_S, pk, \kappa/C)$ is a DDH triple, output the pair (C, K). If neither is the case, then generate a random key K, store (pk_S, pk, C, K) in *EncapList*, and output the pair (C, K).
- On a decapsulation oracle query (pk, C), $\mathcal{B}$ must first check *EncapList* for any previous entries (pk, pk_R, C, K). If such an entry is found, K must be output to maintain consistency. Otherwise, *HashList* is checked for conforming entries (pk, pk_R, κ, K) such that $(pk_S, pk, \kappa/C)$ is a DDH triple, in which case K is returned. If no match is found in either list, then generate a random key K, store (pk, pk_R, C, K) in *EncapList*, and return K.
- Finally, on random oracle queries (κ, pk, pk'), one should first check *HashList* whether the same query has been made before, in which case the same K should be returned. If this is not the case, $\mathcal{B}$ checks whether *EncapList* contains any entries (pk, pk', C, K) such that $(pk, pk', \kappa/C)$ is a DDH triple, in which case K is returned. If no match is found in the list, a random key K is generated and *HashList* is updated accordingly.

Note that it is simple for $\mathcal{B}$ to deal with the flexible oracle queries, since the information about which public keys are in use is embedded in every query. By using the public keys and the supplied DDH oracle, $\mathcal{B}$ is also able to maintain consistency between queries to the three oracles. A new entry is only added to *EncapList* during an oracle query if the corresponding triplet of keys and encapsulation have not been used in a previous query to any of the oracles. Similarly, a new entry is added to *HashList* only if the result has not been fixed (directly or indirectly) previously. Since all new encapsulations and keys are generated by sampling uniformly at random, the variables will also follow the correct distributions.

We now consider what happens when $\mathcal{B}$ plays the IND-CCA2 game for ECISS-KEM with $\mathcal{A}$. As previously stated, $\mathcal{B}$ uses the GDH challenge values g^x andg^y as the public keys pk_S and pk_R, generates *param* from the description of the group, and runs $\mathcal{A}_1$ on $(param, pk_S, pk_R)$, while simulating the oracles as specified. Eventually $\mathcal{A}_1$ terminates, outputting some *state*. To generate a challenge, $\mathcal{B}$ first picks a group element C^* and a key K_0 uniformly at random and adds the value (pk_S, pk_R, C^*, K_0) to *EncapList*. After choosing another random key K_1 and a random bit b, $\mathcal{B}$ runs $\mathcal{A}_2$ on the parameters $(state, C^*, K_b)$. During the execution of $\mathcal{A}_2$, the oracles may be queried as before, with the restriction that the decapsulation query (pk_S, C^*) is forbidden. Eventually $\mathcal{A}_2$ will output some bit b', which is ignored by $\mathcal{B}$. Instead, $\mathcal{B}$ checks whether there are entries in (pk_S, pk_R, C, K) in *EncapList* and (pk_S, pk_R, κ, K) in *HashList* such that $(pk_S, pk_R, \kappa/C)$ is a DDH triple. In this case, κ/C is returned as the solution to the GDH problem; otherwise, a random group element is picked.

Analyzing the advantage of $\mathcal{B}$, we notice that the encapsulation and decapsulation algorithms are simulated perfectly at all times, except during the generation of the challenge C^*. With respect to C^* there are two things that may go wrong; either a

previous oracle query made by $\mathcal{A}_1$ has already fixed a relation between pk_S, pk_R, C^*, and some K or a future encapsulation oracle query by $\mathcal{A}_2$ on pk_R may accidentally reveal the key associated with C^*. Under the assumption that $\mathcal{A}$ is only allowed to make a polynomial number of oracle queries, the probability that either of this happens is negligible and bounded above by $\frac{q_E+q_D}{q}$.

However, because K_0 and K_1 are sampled uniformly at random and independently of C^*, the only other way that $\mathcal{A}$ can learn anything about the value of b is to submit a query (κ^*, pk_S, pk_R) to the random oracle, where $\kappa^*/C^* = g^{sk_S \cdot sk_R} = g^{xy}$. But in this case $\mathcal{B}$ immediately obtains the solution to the Gap Diffie–Hellman problem instance.[3] Hence the advantage of $\mathcal{B}$ in the GDH game will be no worse than that of $\mathcal{A}$. This completes the proof. □

7.4 Hybrid Signcryption with Insider Security

The problem of constructing a framework for insider-secure hybrid signcryption is significantly more complex than the outsider-secure setting, precisely due to the need to protect against insider-specific attacks. We briefly discuss why it appears necessary to use public-key signatures as a starting point, rather than encryption KEMs. Furthermore, we point out the shortcomings of Dent's proposed insider-secure signcryption KEM model [71, 72]. The main focus of the chapter is to present the concept of *signcryption tag-KEMs* [37] and how they avoid the main problems of Dent's model. Examples of schemes that fit the signcryption tag-KEM framework include a modified version of Zheng's signcryption scheme (as described in Sect. 3.3 and 4.3).

7.4.1 From Outsider to Insider Security

While outsider security is sufficient for communication between a trusted set of users, insider security is necessary for more general communication networks, where multiple users who may or may not trust each other wish to communicate in a secure fashion. It is also a necessary (though not sufficient) condition for creating signcryption schemes with non-repudiation functionality [130]—see Sect. 2.2.2. As we saw in Sect. 7.3.3, the model for hybrid signcryption proposed in Sect. 7.3 can never provide insider security, because there is no link between the key encapsulation and the message that is being signcrypted. The logical consequence of this is that any model for an insider-secure signcryption KEM must provide some form

[3] The way the oracles are simulated, $\mathcal{B}$ may also learn the target value from other queries involving pk_S and pk_R, but something other than the challenge.

of integrity service for the message that is being signcrypted, to verify that the relationship between message, key, and encapsulation has not been altered by the adversary. In effect, the encapsulation should provide a signature on all the relevant data for the specific message to be signcrypted, including the public keys of sender and receiver, the encapsulated symmetric key, and the message itself.

Recalling the semantics of a public-key signature scheme and pursuing this idea, it appears reasonable that the encapsulation algorithm should be changed to take the message m as input, as well as the public keys sk_S and pk_R. However, to *verify* the signature on m, it must first be decrypted. Hence it becomes necessary to specify *two* algorithms that are used to unsigncrypt a signcryptext: a decapsulation algorithm to recover the symmetric key K and a verification algorithm to verify that the "signature" part of the encapsulation is valid. On the positive side, it appears reasonable that the security requirement for the DEM can be relaxed to IND-CPA, since the insider-secure signcryption KEM must enforce the integrity of the message anyhow.

Following this intuitive approach yields the original insider-secure signcryption KEMs proposed by Dent [71, 73]. Unfortunately, it is not a particularly pleasant model to work with. One possible reason for this is the fact that it instantiates an example of the "encrypt-and-sign" paradigm, as discussed in Chap. 2 [10]. This means that special considerations have to be taken to avoid information about the signed message leaking through the key encapsulation. Specifically, the security criteria required of the signcryption KEM to provide confidentiality turn out to be quite awkward in Dent's model. To create an indistinguishable signcryption KEM one must consider two separate attack scenarios: one in which the adversary tries to distinguish a real key output by the encapsulation algorithm from a random key (similar to the IND-CCA2 requirement for outsider-secure signcryption KEMs in Sect. 7.3.2) and *another* in which the adversary tries to distinguish between encapsulations of two different messages. In the case of integrity, the standard criterion of strong existential unforgeability (of valid encapsulations) may be applied.

Another flaw of the intuitive approach followed above lies in the proof of the composition theorem for outsider-secure KEM + DEM. In Dent's original proof, the confidentiality of hybrid signcryption relies on the authenticity/integrity of the KEM as well as its confidentiality [71, 72]. This is not very intuitive and leads to poor concrete security: as shown by Bjørstad, the security bound for confidentiality of Zheng's signcryption scheme in the original scheme-specific proof [12, 13] is much tighter than the corresponding proof of security using the functionally equivalent signcryption KEM + DEM formulation [36]. Although Bjørstad suggests an alternate proof of confidentiality avoiding the need for unforgeability, this reimposes the requirement that the DEM must be IND-CCA and is therefore little better in practice. In short, the intuitive definition of an insider-secure signcryption KEM sketched in this section leads to a scheme that does not really simplify the analysis and typically achieves worse security results than a direct proof specific to the scheme under consideration. It follows that a different model is needed to make the concept of insider-secure hybrid signcryption useful.

7.4.2 Signcryption Tag-KEMs

A way to resolve the problems encountered in the previous section appeared in early 2005, when Abe et al. proposed an alternate construction paradigm for hybrid encryption, called *tag-KEMs* [4, 5]. The main idea of tag-KEMs is that the encapsulation algorithm is constructed in two steps: one in which the symmetric key is generated and another where the key is encapsulated in some manner together with an arbitrary string called a *tag*. As we shall see, the security requirement for tag-KEMs also forces the key encapsulation to preserve the integrity of the tag. The authors proceed to present a hybrid construction in which the symmetric ciphertext from the DEM is used as the tag and show that this yields an elegant hybrid encryption scheme where the DEM only needs to be secure against passive attackers (IND-CPA).

It is tempting to adapt the tag-KEM construction paradigm to the insider-secure signcryption setting, precisely because our immediate goal is to design signcryption KEMs that provide integrity services and only require an IND-CPA-secure DEM to make the composition secure. In the hybrid signcryption setting this also acts as an example of the "encrypt-then-sign" paradigm, since the "signature" part of the encapsulation is made on the ciphertext tag instead of on the message itself. It is not unreasonable to expect that such a construction will be more well-behaved under formal analysis, since there is no longer any possibility that the signature component can leak any information about the plaintext to an attacker.[4] Using Abe et al. [4, 5] as inspiration, Bjørstad and Dent [37] give the following formal specification of the tag-KEM construction for signcryption.

Definition 7.9 (**Signcryption tag-KEM**) A *signcryption tag-KEM SCTK* = (`Setup`, $\texttt{KeyGen}_S$, $\texttt{KeyGen}_R$, `Sym`, `Encap`, `Decap`) is defined as a tuple of six algorithms:

- A probabilistic common parameter generation algorithm, `Setup`. It takes as input a security parameter 1^k and returns all the global information *param* needed by users of the scheme, such as choice of groups or hash functions.
- A probabilistic sender key generation algorithm $\texttt{KeyGen}_S$. It takes as input the global information *param* and outputs a public/private keypair (sk_S, pk_S) that is used to send signcrypted messages.
- A probabilistic receiver key generation algorithm $\texttt{KeyGen}_R$. It takes as input the global information *param* and outputs a public/private keypair (sk_R, pk_R) that is used to receive signcrypted messages.

[4] The alternate "sign-then-encrypt" construction might be even more appealing, because it keeps the formal signature where it logically and semantically belongs: on the plaintext. However, it does not appear to be practical to build a model for hybrid signcryption schemes instantiating this concept, due to the need to divide the signcryption KEM into separate "signature" and "encapsulation" parts, and the complex information flows resulting from this.

- A probabilistic symmetric key generation algorithm Sym. It takes as input the private key of the sender sk_S and the public key of the receiver pk_R and outputs a symmetric key K together with internal state information ω.
- A probabilistic key encapsulation algorithm Encap. It takes as input some state information ω and an arbitrary tag τ, and returns an encapsulation C.[5]
- A deterministic decapsulation and verification algorithm Decap. It takes as input the sender's public key pk_S, the receiver's private key sk_R, an encapsulation C, and a tag τ. The algorithm returns either a symmetric key K or the unique error symbol $\perp$.

By combining the above signcryption tag-KEM with a DEM, we obtain a hybrid signcryption scheme as follows.

Definition 7.10 (**SCTK+DEM hybrid signcryption scheme**) Suppose that (Setup, KeyGen_S, KeyGen_R, Sym, Encap, Decap) is a signcryption tag-KEM and (Enc, Dec) a DEM and that the keys produced by the signcryption tag-KEM are of appropriate length for use with the DEM for all security parameters k. Then we can construct a hybrid signcryption scheme by using the Setup, KeyGen_S, and KeyGen_R from the SCTK and defining the algorithms Signcrypt and Unsigncrypt as follows.

- The Signcrypt algorithm takes as input the private key of the sender sk_S, the public key of the receiver pk_R, and a message m. It performs the following steps:
  1. Compute $\text{Sym}(sk_S, pk_R)$ to obtain a symmetric key K and state information ω.
  2. Compute $\text{Enc}_K(m)$ to produce a ciphertext C_2.
  3. Compute $\text{Encap}(\omega, C_2)$, using C_2 as the tag τ to produce the ciphertext C_1.
  4. Output the signcryptext $C \leftarrow (C_1, C_2)$ and halt.
- The Unsigncrypt algorithm takes as input the public key of the sender pk_S, the private key of the receiver sk_R, and a ciphertext C. It performs the following steps:
  1. Parse C to obtain its component parts C_1 and C_2.
  2. Compute $K \leftarrow \text{Decap}(pk_S, sk_R, C_1, C_2)$, using C_1 as the encapsulation and C_2 as the tag.
  3. If Decap returned $\perp$, output $\perp$, and halt. Otherwise, compute $m \leftarrow \text{Dec}_K(C_2)$.
  4. Output m and halt.

[5] In principle, this algorithm can always be represented as a deterministic algorithm, which takes as input the appropriate amount of random bits embedded in ω as a string. In practice this is often the case, as random nonces may be chosen as part of Encap and used to create a random K, and then passed along to Sym as part of ω. However, from a theoretical point of view, if Encap is only *expected* polynomial time, the deterministic version will have an (arbitrarily small) probability of failing.

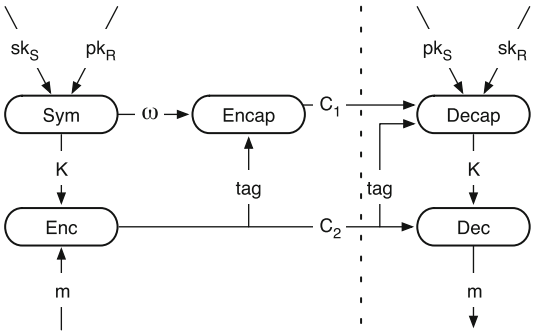

Fig. 7.5 Data flow in the insider-secure signcryption tag-KEM + DEM construction

The data flow between the `Signcrypt` and `Unsigncrypt` algorithms is illustrated in Fig. 7.5. Notice in particular how the symmetric ciphertext C_2 is used as the "tag" input to both `Encap` and `Decap`.

As it turns out, it is quite possible to express Zheng's signcryption scheme as a signcryption tag-KEM + DEM construction. However, it requires the trivial alteration of having the "signature" part of the scheme act on the symmetric ciphertext instead of the message itself. The resulting scheme is essentially the scheme of Gamage et al.—see Sect. 4.3.3. This is not expected to have any effect on the overall security of the scheme, an assumption that is verified independently by Bjørstad and Dent [37]. A concrete specification of the "Zheng signcryption tag-KEM" is given in Fig. 7.6. Since Zheng's scheme is known to be secure (in the random oracle model) [12, 13] this yields confidence that the signcryption tag-KEM construction is viable and useful, provided that a good generic security reduction can be made. As we shall see in Sects. 7.4.3 and 7.4.4 this is indeed the case.

7.4.3 Security Criteria for Signcryption Tag-KEMs

For the signcryption tag-KEM construction to be viable, we need clear and well-defined notions of what it means for a signcryption tag-KEM to be secure. Furthermore, these notions must be useful by themselves, so that it is possible to prove that suggested signcryption tag-KEMs fulfill them and admit an efficient security reduction for the generic hybrid signcryption scheme obtained by combining an SCTK with a DEM. As we observed in Sect. 7.4.1, this is not always achievable. However, in the signcryption tag-KEM setting we find intuitive and simple notions of security for both confidentiality and authenticity/integrity. We will define these security notions analogously with the main definitions given in Chap. 3, specifically the multi-user outsider model for confidentiality and the multi-user insider model for unforgeability. (Extensions to the other models given in Chap. 3 can be simply made using the techniques in this section.)

$\texttt{Setup}(1^k)$
Pick a random large prime p with
a k-bit prime q dividing $p-1$.
Pick $g \in \mathbb{Z}_p^*$ of order q.
Pick cryptographic hash functions:
$G : \{0,1\}^* \to \mathcal{K}$
$H : \{0,1\}^* \to \mathbb{Z}_q$
$param \leftarrow (p,q,g,G,H)$
Return $param$

$\texttt{KeyGen}_S(param)$
$x_S \xleftarrow{R} \mathbb{Z}_q$; $y_S \leftarrow g^{x_S}$
$sk_S \leftarrow (x_S, y_S)$; $pk_S \leftarrow y_S$
Return (sk_S, pk_S)

$\texttt{KeyGen}_R(param)$
$x_R \xleftarrow{R} \mathbb{Z}_q$; $y_R \leftarrow g^{x_R}$
$sk_R \leftarrow (x_R, y_R)$; $pk_R \leftarrow y_R$
Return (sk_R, pk_R)

$\texttt{Sym}(param, sk_S, pk_R)$
Parse sk_S as (x_S, y_S); Parse pk_R as y_R
If $y_R \notin \langle g \rangle \setminus \{1\}$ then return $\perp$
$x \xleftarrow{R} \mathbb{Z}_q$; $\kappa \leftarrow y_R^x$; $K \leftarrow G(\kappa)$
$bind \leftarrow pk_S || pk_R$; $\omega \leftarrow (x_S, x, \kappa, bind)$
Return (K, ω)

$\texttt{Encap}(\omega, \tau)$
Parse ω as $(x_S, x, \kappa, bind)$
$r \leftarrow H(\tau || bind || \kappa)$
If $r + x_S = 0$ then return $\perp$
$s \leftarrow x/(x_S + r)$
$C \leftarrow (r,s)$
Return C

$\texttt{Decap}(param, pk_S, sk_R, C, \tau)$
Parse sk_R as (x_R, y_R); Parse pk_S as y_S
If $y_S \notin \langle g \rangle \setminus \{1\}$ then return $\perp$
Parse C as (r,s)
If $r \notin \mathbb{Z}_q$ or $s \notin \mathbb{Z}_q$
Return $\perp$
$\kappa \leftarrow (y_S g^r)^{s x_R}$; $K \leftarrow G(\kappa)$
$bind \leftarrow pk_S || pk_R$
If $H(\tau || bind || \kappa) = r$ then return K
Else return $\perp$

Fig. 7.6 A complete specification of the Zheng signcryption tag-KEM (Zheng-SCTK). This scheme should be compared with the Gamage et al. specification in Sect. 4.3.3

A signcryption tag-KEM maintains confidentiality when it is impossible for an adversary to distinguish whether a given key K is embedded in an encapsulation C or not. This is the only requirement needed. However, as the adversary is allowed to specify the tag and may access flexible oracles, this is sufficient to ensure that the encapsulation is not malleable with respect to the tag. Since the symmetric key generation and encapsulation algorithms do not receive the unencrypted plaintext as input, the additional requirement of input indistinguishability is not necessary. For a given security parameter k, the IND-CCA2 game between challenger and a three-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ runs as follows:

1. The challenger generates a set of global information $param \xleftarrow{R} \texttt{Setup}(1^k)$ and the key pairs $(sk_S, pk_S) \xleftarrow{R} \texttt{KeyGen}_S(param)$ and $(sk_R, pk_R) \xleftarrow{R} \texttt{KeyGen}_R(param)$ for the sender and the receiver.
2. The adversary runs $\mathcal{A}_1$ on the input $(param, pk_S, pk_R)$. During its execution, $\mathcal{A}_1$ is given access to flexible symmetric key generation, encapsulation, and decapsulation oracles:

   - The symmetric key generation oracle $\mathcal{O}_{Sym}$ takes a public key pk as input and runs $(K, \omega) \xleftarrow{R} \texttt{Sym}(sk_S, pk)$. It then stores the value of ω, hidden from the

view of the adversary, and overwriting any previous value. The oracle outputs the key K.
- The key encapsulation oracle $\mathcal{O}_{Encap}$ takes a tag τ as input and checks whether there is a stored ω. If there is not, it outputs $\perp$. Otherwise it erases the value of ω from storage, computes $\texttt{Encap}(\omega, \tau)$, and outputs the result.
- The decapsulation/verification oracle $\mathcal{O}_{Decap}$ takes a public sending key pk, an encapsulation C, and a tag τ as input. It then computes $\texttt{Decap}(pk, sk_R, C, \tau)$ and outputs the result.

$\mathcal{A}_1$ terminates by outputting some state information $state_1$.

3. The challenger computes $(K_0, \omega^*) \xleftarrow{R} \texttt{Sym}(sk_S, pk_R)$, generates a random symmetric key $K_1 \xleftarrow{R} \mathcal{K}$, where $\mathcal{K}$ is the output keyspace of the tag-KEM, and a random bit $b \xleftarrow{R} \{0, 1\}$.
4. The adversary runs $\mathcal{A}_2$ on the input $(state_1, K_b)$. During its execution, $\mathcal{A}_2$ may query the oracles as before. $\mathcal{A}_2$ terminates by outputting an arbitrary tag τ^* as well as any necessary state information $state_2$.
5. The challenger computes the challenge encapsulation $C^* \xleftarrow{R} \texttt{Encap}\ (\omega^*, \tau^*)$.
6. The adversary runs $\mathcal{A}_3$ on the input $(C^*, state_2)$. During its execution, $\mathcal{A}_2$ may query the same oracles as before, with the restriction that (pk_S, C^*, τ^*) is not a valid query to the decapsulation oracle. $\mathcal{A}_3$ terminates by outputting a guess b' for the value of b.

The adversary wins the game if it is successful at guessing the hidden bit, i.e., $b = b'$. The advantage of $\mathcal{A}$ is defined as $|Pr[b = b'] - 1/2|$.

Definition 7.11 (**Indistinguishable signcryption tag-KEM**) A signcryption tag-KEM is said to be (multi-user outsider) indistinguishable (IND-CCA2) secure if the advantage of any polynomial-time adversary $\mathcal{A}$ in the IND-CCA2 game is negligible with respect to the security parameter k.

It is important to note the behind-the-scenes interaction between the symmetric key generation and encapsulation oracles in the IND-CCA2 game. This is done in order to let the adversary perform *completely* adaptive encapsulations without having access to the state information stored in ω (which may include nonces, private keys, random coins, and other information strictly internal to the execution of the signcryption tag-KEM).

With respect to the authenticity and integrity of signcryption tag-KEMs, we adapt the usual notion of strong existential unforgeability. The precise requirement is that an adversary should not be able to find encapsulation/tag pairs (C, τ) under some sender's key pk such that $\perp \neq \texttt{Decap}(pk, sk_R, C, \tau)$. We let the adversary choose the receiving entity to which the adversary wishes to forge messages. The attack game corresponding to the sUF-CMA security of a signcryption tag-KEM runs as follows, for a given security parameter k:

1. The challenger generates a set of global information $param \xleftarrow{R} \texttt{Setup}(1^k)$ and a sender keypair $(sk_S, pk_S) \xleftarrow{R} \texttt{KeyGen}_S(param)$.

2. The adversary $\mathcal{A}$ is run on the input $(param, pk_S)$. During its execution, $\mathcal{A}$ has access to oracles for symmetric key generation and encapsulation corresponding to the sender's private key sk_S, as defined previously. $\mathcal{A}$ terminates by outputting a fixed receiver keypair (sk_R, pk_R), an encapsulation C, and a tag τ.

The adversary wins the game if $\perp \neq \texttt{Decap}(pk_S, sk_R, C, \tau)$, provided that the encapsulation oracle never returned C when queried with the tag τ and the ω loaded from storage was not the result of a symmetric key oracle query on pk_R. The advantage of $\mathcal{A}$ is simply the probability Pr[$\mathcal{A}$ wins].

Definition 7.12 (**Unforgeable signcryption tag-KEM**) A signcryption tag-KEM is said to be (multi-user insider) strongly unforgeable (sUF-CMA secure) if the advantage of any polynomial-time adversary $\mathcal{A}$ in the sUF-CMA game is negligible with respect to the security parameter k.

Definition 7.13 (**Secure signcryption tag-KEM**) A signcryption tag-KEM is said to be *secure* if it is indistinguishable and unforgeable.

7.4.4 Security of the SCTK+DEM Construction

It remains to show that the combination of a secure signcryption tag-KEM and a secure DEM indeed yields a secure signcryption scheme. Although the original paper on signcryption tag-KEMs only investigated this in the two-user (ADR) model [37], later work has extended this to the multi-user (BSZ) model as well [200].

Theorem 7.3 *(Security of SCTK + DEM construction) Let SC be a hybrid signcryption scheme constructed from a signcryption tag-KEM and a DEM. If the signcryption tag-KEM is IND-CCA2 secure (Definition 7.9) and the DEM is IND-CPA secure, then SC is multi-user outsider FSO/FUO-IND-CCA2 secure (Definition 3.1) with the bound*

$$\varepsilon_{SC,IND\text{-}CCA2} \leq 2\,\varepsilon_{SCTK,IND\text{-}CCA2} + \varepsilon_{DEM,IND\text{-}CPA} \tag{7.5}$$

Furthermore, if the signcryption tag-KEM is sUF-CMA secure (Definition 7.12), then SC is multi-user insider FSO/FUO-sUF-CMA secure (Definition 3.2) with the bound

$$\varepsilon_{SC,sUF\text{-}CMA} \leq \varepsilon_{SCTK,sUF\text{-}CMA} \tag{7.6}$$

Proof We begin by proving the indistinguishability of the construction. The proof uses standard techniques and has a similar approach as the corresponding proof of security for encryption tag-KEMs [4, 5] and the proof of Theorem 7.1.

Let G_0 be the regular FSO/FUO-IND-CCA2 game for multi-user-secure signcryption, as described in Chap. 3. We modify G_0 so that the hybrid signcryption procedure uses a key drawn uniformly at random when computing the challenge signcryptext, instead of the actual key output by `Sym`. The resulting game is referred

$\mathcal{D}_1(param, pk_S, pk_R; \mathcal{O}_{Sym}, \mathcal{O}_{Encap}, \mathcal{O}_{Decap})$:
$(m_0, m_1, s) \xleftarrow{R} \mathcal{A}_1(param, pk_R; \mathcal{O}_{SC}, \mathcal{O}_{USC})$.
$state_1 \leftarrow (param, pk_S, pk_R, m_0, m_1, s)$.
Return $(state_1)$.

$\mathcal{D}_2(K^*, state_1; \mathcal{O}_{Sym}, \mathcal{O}_{Encap}, \mathcal{O}_{Decap})$:
$b \xleftarrow{R} \{0,1\}$.
$C_2^* \leftarrow \mathtt{Enc}_K(m_b)$.
$state_2 \leftarrow (state_1, b, C_2^*)$.
Return $(C_2^*, state_2)$.

$\mathcal{D}_3(C_1^*, state_2; \mathcal{O}_{Decap})$:
Parse $state_2$ as
$((param, pk_S, pk_R, m_0, m_1, s), b, C_2^*)$.
$C^* \leftarrow (C_1^*, C_2^*)$.
$b' \xleftarrow{R} \mathcal{A}_2(C^*, s; \mathcal{O}_{SC}, \mathcal{O}_{USC})$.
If $b = b'$, return 1.
Else return 0.

$\mathcal{O}_{SC}(pk, m)$:
$K \xleftarrow{R} \mathcal{O}_{Sym}(pk)$.
$C_2 \leftarrow \mathtt{Enc}_K(m)$.
$C_1 \xleftarrow{R} \mathcal{O}_{Encap}(C_2)$.
$C \leftarrow (C_1, C_2)$.
Return C.

$\mathcal{O}_{USC}(pk, C)$:
Parse C as (C_1, C_2).
If $\perp = \mathcal{O}_{Decap}(pk, C_1, C_2)$ then
Return $\perp$ and halt.
Else $K \leftarrow \mathcal{O}_{Decap}(pk, C_1, C_2)$.
$m \leftarrow \mathtt{Dec}_K(C_2)$.
Return m.

Fig. 7.7 A complete specification of the distinguisher algorithm $\mathcal{D}$ for the SCTK

to as G_1. Let X_0 and X_1 be the events that some adversary $\mathcal{A}$ guesses the correct key in G_0 and G_1, respectively. We bound $|\Pr[X_1] - \Pr[X_0]| \leq 2\,\varepsilon_{SCTK,IND\text{-}CCA2}$ by constructing a distinguisher algorithm $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3)$ that uses $\mathcal{A}$ to win the IND-CCA2 game against the underlying signcryption tag-KEM and applying Lemma 7.1. As can be seen from the specification in Fig. 7.7, $\mathcal{D}$ simulates the environment of $\mathcal{A}$ perfectly, playing either G_0 or G_1 depending on the hidden bit (which $\mathcal{D}$ is trying to find). This is precisely what is needed to apply Lemma 7.1.

A notable difference from the proof of Theorem 7.1 is that the random key introduced in G_1 is not needed to unsigncrypt oracle queries on the form $\big(pk_R, (C_1^*, C_2)\big)$. This is because of the way decapsulation works, where both the encapsulation and the tag must have an effect on the key.

Finally, the advantage of $\mathcal{A}$ in G_1 is easily seen to be the same as that of an adversary $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ performing a passive attack on the DEM. Such an adversary is specified in Fig. 7.8. We note that $\mathcal{B}$ wins the IND-CPA game by distinguishing whether the challenge ciphertext (which from the view of $\mathcal{B}$ has been encrypted with an unknown random key K) if and only if $\mathcal{A}$ can distinguish the correct signcryptext which it is being wrapped into by $\mathcal{B}$. A major difference from the proof of Theorem 7.1 is that the adversary is no longer *able* to make chosen ciphertext queries under the symmetric key used for the challenge, which is why we get away with passive (IND-CPA) security. This completes the proof of confidentiality.

Demonstrating the unforgeability of *SC* relative to the corresponding signcryption tag-KEM is simpler yet. Any valid forgery of *SC* requires that the adversary comes up with an encapsulation C_1 that acts as a signature on the ciphertext C_2. In the language of the signcryption tag-KEM, the adversary must in some way have constructed an encapsulation that acts as a signature on the ciphertext tag. This is precisely what it means to break the sUF-CMA security of a signcryption tag-KEM. Figure 7.9 gives the formal specification of an adversary $\mathcal{B}'$ that is able

$\mathcal{B}_1(1^k)$:
$param \xleftarrow{R} \mathtt{Setup}(1^k)$.
$(sk_S, pk_S) \xleftarrow{R} \mathtt{KeyGen}_S(param)$.
$(sk_R, pk_R) \xleftarrow{R} \mathtt{KeyGen}_R(param)$.
$(m_0, m_1, s) \xleftarrow{R} \mathcal{A}_1(param, pk_S, pk_R; \mathcal{O}_{SC}, \mathcal{O}_{USC})$.
$state \leftarrow (param, sk_S, pk_S, sk_R, pk_R, m_0, m_1, s)$.
Return $(m_0, m_1, state)$.

$\mathcal{B}_2(C_2^*, state)$:
Parse *state* as
$(param, sk_S, pk_S, sk_R, pk_R, m_0, m_1, s)$.
$(K, \omega) \xleftarrow{R} \mathtt{Sym}(sk_S, pk_R)$.
$C_1^* \xleftarrow{R} \mathtt{Encap}(\omega, C_2^*)$.
$C^* \leftarrow (C_1^*, C_2^*)$.
$b \xleftarrow{R} \mathcal{A}_2(s, C^*; \mathcal{O}_{SC}, \mathcal{O}_{USC})$.
Return b.

$\mathcal{O}_{SC}(pk, m)$:
$(K, \omega) \xleftarrow{R} \mathtt{Sym}(pk, sk_R)$.
$C_2 \leftarrow \mathtt{Enc}_K(m)$.
$C_1 \xleftarrow{R} \mathtt{Encap}(\omega, C_2)$.
$C \leftarrow (C_1, C_2)$.
Return C.

$\mathcal{O}_{USC}(pk, C)$:
Parse C as (C_1, C_2).
If $\perp \leftarrow \mathtt{Decap}(pk, sk_R, C_1, C_2)$ then
Return $\perp$ and halt.
Else $K \leftarrow \mathtt{Decap}(pk, sk_R, C_1, C_2)$.
$m \leftarrow \mathtt{Dec}_K(C_2)$.
Return m.

Fig. 7.8 A complete specification of the distinguisher algorithm $\mathcal{B}$ for the DEM

$\mathcal{B}'(param, pk_S; \mathcal{O}_{Sym}, \mathcal{O}_{Encap})$:
$(sk_R, pk_R, m, C) \xleftarrow{R} \mathcal{A}(param, pk_S; \mathcal{O}_{SC})$.
Parse C as (C_1, C_2).
Return (sk_R, pk_R, C_1, C_2).

$\mathcal{O}_{SC}(pk, m)$:
$K \xleftarrow{R} \mathcal{O}_{Sym}(pk)$.
$C_2 \leftarrow \mathtt{Enc}_K(m)$.
$C_1 \xleftarrow{R} \mathcal{O}_{Encap}(C)$.
$C \leftarrow (C_1, C_2)$.
Return C.

Fig. 7.9 A complete specification of the forgery algorithm $\mathcal{B}'$ for the SCTK

to forge encapsulations for the signcryption tag-KEM, given a forger for the hybrid signcryption scheme SC.

To verify that $\mathcal{B}'$ indeed constitutes an efficient forgery algorithm, we note that it outputs a valid encapsulation (i.e., something that decapsulates to some message $m \neq \perp$) whenever $\mathcal{A}$ has outputted a valid forgery of SC. It is also a simple observation that $\mathcal{B}'$ simulates the runtime environment of $\mathcal{A}$ perfectly, since it does not make any independent actions and simply passes along oracle queries to the signcryption tag-KEM oracles.

The only remaining requirement is that the value C_1 had not been the end result of any pair of oracle queries $\mathcal{O}_{Sym}(pk_R)$ and $\mathcal{O}_{Encap}(C_2)$. The corresponding requirement of $\mathcal{A}$ is that the oracle simulated by $\mathcal{O}_{SC}$ did not respond with C on a query of (pk_R, m). Since decapsulation is deterministic, we note that C was only returned by $\mathcal{O}_{SC}$ if m was part of the query. Furthermore, $\mathcal{O}_{SC}$ will only respond with C if C_1 and C_2 were the output and input to $\mathcal{O}_{Encap}$. Finally, the public key pk_R is passed directly through $\mathcal{O}_{SC}$, so the only time it will be part of a query to $\mathcal{O}_{Sym}$ is if it was part of the signcryption oracle query from $\mathcal{A}$. We conclude that the two algorithms $\mathcal{A}$ and $\mathcal{B}'$ have identical advantages in their respective games. This completes the proof. □

The proof of Theorem 7.3 also holds in alternate security models, from the two-user (ADR) models specified in Chap. 2 [37] and up to the full multi-user security

models used here. It may also readily be extended to a general multi-user insider setting, where the adversary is allowed to pick the sending keys used in the indistinguishability game. As long as the same notion of security is applied to both the hybrid signcryption scheme and the underlying signcryption tag-KEM, the general reductions remain valid with the appropriate modifications.

7.4.5 Insider-Secure Hybrid Signcryption in Practice

As we have established, an insider-secure signcryption KEM needs to combine the key encapsulation functionality of a regular KEM with the authenticity-preserving and integrity-preserving features of a digital signature scheme. From prior experience with other hybrid schemes, an obvious approach would be to take a hybrid encryption scheme as a starting point and extend it to fulfill our additional requirements. However, this is harder than it seems. In fact, apart from trivial compositions combining KEMs and signature schemes, there are no known insider-secure hybrid signcryption schemes based on encryption KEMs.

The opposite approach is to start with a secure signature scheme and tweak it in such a way that it also acts as a KEM. One way to do this is to alter the computation of some internal value in such a way that it depends on the keys of both sender and receiver and using it to derive a symmetric key. This method has been more successful, with Zheng's original scheme [203] being the canonical example. But there is no general method known to generate efficient hybrid signcryption schemes from *arbitrary* signature schemes. However, most known insider-secure hybrid signcryption schemes apply exactly the same trick as Zheng's scheme (see Chaps. 4, 5, and 6).

The idea used by Zheng and the others applies to signature schemes that work in a very specific manner: to sign, one must pick a random nonce n, use it to compute a random group element g^n, which is hashed together with the message to be signed, whereupon some computations on the result based on the signer's private key are performed. To verify, the public key of the signer is used to reconstruct g^n from the signature data and verify that the output of the hash is correct. A signcryption scheme can therefore be created by modifying the first step to compute the randomizer as pk_S^n, requiring sk_S to reconstruct it from the g^n computed during normal verification. Although it has not been proven, it is conjectured that this construction works in general; all that is currently known is that it is secure (in the random oracle model) in several specific cases.

However, other efficient methods of constructing efficient insider-secure hybrid signcryption schemes from scratch are not known, and existing schemes that fit into the signcryption tag-KEM model are all based on variants of the Diffie–Hellman problem (see Chaps. 4 and 5). The scheme proposed by Malone-Lee (see Sect. 4.6.2) is of particular interest as it is an example of a hybrid signcryption scheme with non-repudiation, while the schemes proposed by Bjørstad and Dent (see Sect. 4.7) and by Libert and Quisquater (see Sect. 5.5) are of interest as they have particularly tight security reductions.

A rather different instantiation of signcryption tag-KEMs can be made by extending ordinary (non-hybrid) signcryption schemes that support transmission of associated plaintext data together with the signcryption [167]. In these schemes, the signcryption algorithm "binds" the associated data to the signcryptext, providing integrity protection for both. As we have seen previously in this chapter, this is *exactly* what we want. Intuitively, one may think of the plaintext label as the tag and use the regular signcryption scheme to signcrypt a symmetric key.

This construction is useful for precisely the same reasons that makes hybrid signcryption so appealing in the first place: it removes the restriction of non-hybrid schemes to small message spaces, which make them inefficient and slow for long messages. A hybrid construction using schemes with associated data was first suggested by Dodis et al. in [77], building on the theory of concealment schemes (discussed further in Chap. 8). Bjørstad and Dent [37] prove that signcryption tag-KEMs built in this manner yield the same scheme.

Formally, the syntax of a signcryption scheme with associated data differs from usual only in the ways the associated data are handled:

- The `Signcrypt` algorithm takes as additional input the associated data d, so that the syntax becomes $C \xleftarrow{R} \texttt{Signcrypt}(sk_S, pk_R, m, d)$.
- The `Unsigncrypt` algorithm also requires d as part of its input, hence $m \leftarrow \texttt{Unsigncrypt}(pk_S, sk_R, C, d)$.
- Signcryption and unsigncryption oracles are modified accordingly, so that the adversary may choose the value of d (or leave it empty) when making oracle queries.

The security criteria are also altered in the obvious manner, to ensure that the integrity of the associated data is maintained. With these alterations in mind, the construction of a signcryption tag-KEM is straightforward:

- The `Sym` algorithm takes sender and receiver keys sk_S and pk_R as input. It picks a symmetric key K uniformly at random, sets $\omega \leftarrow (sk_S, pk_R, K)$, and returns the pair (ω, K).
- The `Encap` algorithm takes state information ω and a tag τ as input. It parses $(sk_S, pk_R, K) \leftarrow \omega$, computes $C \xleftarrow{R} \texttt{Signcrypt}(sk_S, pk_R, K, \tau)$, and returns C.
- The `Decap` algorithm takes keys pk_S and sk_R, the signcryptext C, and tag τ as input. It uses the computes $K \leftarrow \texttt{Unsigncrypt}(pk_S, sk_R, C, \tau)$ and returns K.

It is quite straightforward to show that this construction is secure, and a proof will not be given here. The main intuition is that the signcryption tag-KEM acts as a wrapper for the underlying signcryption scheme in such a way that an adversary has very little opportunity to do anything "interesting." In the random oracle model, signcryption schemes using the common "hash-and-sign" approach can often be used in this manner by using the plaintext label as an additional input to the hash function.