Building Better Signcryption Schemes with Tag-KEMs

Tor E. Bjørstad[1] and Alexander W. Dent[2]

[1] The Selmer Center, Department of Informatics,
University of Bergen, Norway
[2] Royal Holloway, University of London,
Egham, Surrey, U.K.
`tor.bjorstad@ii.uib.no` `a.dent@rhul.ac.uk`

Abstract. Signcryption schemes aim to provide all of the advantages of simultaneously signing and encrypting a message. Recently, Dent [11, 12] and Bjørstad [5] investigated the possibility of constructing provably secure signcryption schemes using hybrid KEM-DEM techniques [10]. We build on this work by showing that more efficient insider secure hybrid signcryption schemes can be built using tag-KEMs [1]. To prove the effectiveness of this construction, we will provide several examples of secure signcryption tag-KEMs, including a brand new construction based on the Chevallier-Mames signature scheme [8] which has the tightest known security reductions for both confidentiality and unforgeability.

1 Introduction

The signcryption primitive was introduced by Zheng in 1997 [17] to study asymmetric schemes that offer most or all the benefits provided by public-key encryption and signature schemes. Signcryption schemes must provide message authenticity, confidentiality and integrity, and may also offer a way to provide non-repudiation. As such, a signcryption scheme provides a secure, authenticated channel for message transmission. Although Zheng only considered schemes that are more computationally efficient than a direct composition of encryption and signature schemes, the definition of signcryption is normally expanded to include any asymmetric scheme that provides this functionality, regardless of efficiency. Direct composition of public-key encryption and signatures has been studied by An *et.al.* [2].

In order to obtain efficient encryption schemes in practice, hybrid techniques are commonly used. The practice of combining symmetric and asymmetric schemes to encrypt and transmit long messages efficiently has been common knowledge for many years. However, formal analysis was first performed by Cramer and Shoup in the late 1990s [10]. The usual construction paradigm, known as the KEM-DEM construction, consists of two parts: a key encapsulation mechanism (KEM) and a data encapsulation mechanism (DEM). The KEM uses asymmetric techniques to encrypt a symmetric key, while the DEM uses a symmetric cipher to encrypt the message payload using the key from the KEM.

The main benefit of the KEM-DEM construction paradigm is that the security of KEM and DEM may be analyzed separately.

The use of hybrid techniques to build signcryption schemes has been studied by Dent [11–13] and Bjørstad [5]. This has provided a useful perspective for analysis of those classes of signcryption schemes that use hybrid techniques. However, previous efforts have yielded complex verification-decryption (unsigncryption) algorithms, stemming from the need to verify a link between message, key and encapsulation. This article will examine a way to simplify the hybrid construction through use of tag-KEMs [1]. We show that adapting the tag-KEM + DEM construction to signcryption yields simpler scheme descriptions and better generic security reductions than previous efforts.

To demonstrate the usefulness of this new paradigm, we construct several signcryption schemes based on signcryption tag-KEMs. The first is a simple modification of Zheng's original signcryption scheme [17]. This scheme has become baseline standard for judging the efficiency and security of any new signcryption scheme or construction method. The second is a new signcryption scheme based on the Chevallier-Mames signature scheme [8]. As far as the authors are aware, this new signcryption scheme has the tightest known security bounds.

2 Preliminaries

2.1 Signcryption

The signcryption primitive was introduced in 1997 by Zheng [17].

Definition 1 (Signcryption). *A signcryption scheme* $\mathcal{SC} = (Com, Key_S, Key_R, SC, USC)$ *is defined as tuple of five algorithms.*

- *A probabilistic common parameter generation algorithm, Com. It takes as input a security parameter 1^k, and returns all the global information I needed by users of the scheme, such as choice of groups or hash functions.*
- *A probabilistic sender key generation algorithm, Key_S. It takes as input the global information I, and outputs a private/public keypair (sk_S, pk_S) that is used to send signcrypted messages.*
- *A probabilistic receiver key generation algorithm, Key_R. It takes as input the global information I, and outputs a private/public keypair (sk_R, pk_R) that is used to receive signcrypted messages.*
- *A probabilistic signcryption algorithm SC. It takes as input the private key of the sender sk_S, the public key of the receiver pk_R, and a message m. It outputs a signcryptext σ.*
- *A deterministic unsigncryption algorithm USC. It takes as input the public key of the sender pk_S, the private key of the receiver sk_R, and a signcryptext σ. It outputs either a message m or the unique error symbol $\perp$.*

For a signcryption scheme to be sound, it is required that $m = USC\big(pk_S, sk_R, SC(sk_S, pk_R, m)\ \big)$ *for (almost) all fixed keypairs* (sk_S, pk_S) *and* (sk_R, pk_R).

For a signcryption scheme to be useful, it is necessary that it also satisfies well-defined notions of security corresponding to the design goals of confidentiality and authenticity/integrity. Formally, the probability of an adversary breaking the security of signcryption should be *negligible* as a function of the security parameter 1^k.

Definition 2 (Negligible function). *A function $f : \mathbb{N} \rightarrow \mathbb{R}$ is negligible if for every polynomial p over $\mathbb{N}$, there exists a $n_0 \in \mathbb{N}$ such that $|f(n)| \leq \frac{1}{|p(n)|}$ for all $n \geq n_0$.*

Security models are commonly phrased in terms of games played between a hypothetical *challenger* and an *adversary*, who are both modelled as probabilistic Turing machines. In a game, the adversary is challenged to defeat a certain well-defined aspect of the underlying scheme's security under controlled circumstances. As long as the adversary's advantage (with respect to random guessing) at winning the game is negligible, the scheme may be considered to be secure.

The canonical notion of confidentiality for signcryption is that of indistinguishability of signcryptions. This is adapted directly from the corresponding security notion for encryption schemes: an adversary should not, even when given adaptive access to signcryption and unsigncryption oracles, be able to distinguish between the signcryption of two messages of his own choice. Indistinguishability of signcryptions with respect to an adaptive chosen ciphertext adversary is commonly referred to as IND-CCA2. This security notion may be expressed by a game played between the challenger and a two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. For a given security parameter 1^k, the game proceeds as follows:

1. The challenger generates a set of global parameters $I = Com(1^k)$, a sender keypair $(sk_S, pk_S) = Key_S(I)$ and a receiver keypair $(sk_R, pk_R) = Key_R(I)$.
2. The adversary runs $\mathcal{A}_1$ on the input (I, pk_S, pk_R). During its execution, $\mathcal{A}_1$ is given access to signcryption and unsigncryption oracles. The signcryption oracle takes a message m as input, and returns $SC(sk_S, pk_R, m)$. The unsigncryption oracle takes a signcryptext σ as input, and returns $USC(pk_S, sk_R, \sigma)$. $\mathcal{A}_1$ terminates by outputting two messages (m_0, m_1) of equal length, and some state information *state*.
3. The challenger computes a challenge signcryption by generating a random bit $b \in \{0, 1\}$ and computing $\sigma = SC(sk_S, pk_R, m_b)$.
4. The adversary runs $\mathcal{A}_2$ on the input $(state, \sigma)$. During its execution, $\mathcal{A}_2$ has access to signcryption and unsigncryption oracles as above, with the restriction that the challenge signcryptext σ may not be asked to the unsigncryption oracle. $\mathcal{A}_2$ terminates by outputting a guess b' for the value of b.

The adversary wins the game whenever $b = b'$. The advantage of $\mathcal{A}$ is defined as $\big|Pr[b = b'] - 1/2\big|$.

With regards to the authenticity and integrity of signcryption, the notion of existential forgery (UF-CMA) is adapted from analysis of signature schemes. It is however necessary to distinguish between different types of such forgery. In an *outsider-secure* signcryption scheme, the adversary is given access to signcryption and unsigncryption oracles, and the public keys of the sender and receiver. For the stronger notion of *insider security*, the unsigncryption oracle is replaced by giving the adversary direct access to the receiver's *private* key. This article will focus on insider-secure signcryption only. Simple and efficient hybrid signcryption schemes secure against outsiders have been constructed by Dent [13].

It is also necessary to specify what it means for the adversary to win the security game. Traditionally, the requirement has been that the adversary should output a message/signcryptext pair where the message has not been asked to the signcryption oracle. This reflects the "business use" of a signature, where an attacker's ability to produce a new signature on a previously signed message does not constitute a security risk. A stronger notion is that of *strong* existential unforgeability (sUF-CMA). In this case, the only restriction is that the returned signcryptext was not returned by the signcryption oracle when queried on the submitted message. Given a security parameter 1^k, a game for the sUF-CMA insider security of a signcryption scheme procceds as follows:

1. The challenger generates a set of global parameters $I = Com(1^k)$, a sender keypair $(sk_S, pk_S) = Key_S(I)$ and a receiver keypair $(sk_R, pk_R) = Key_R(I)$.
2. The adversary $\mathcal{A}$ is run on the input (I, pk_S, sk_R, pk_R). During its execution, $\mathcal{A}$ is given access to a signcryption oracle, which takes a message m as input and returns $SC(sk_S, pk_R, m)$. $\mathcal{A}$ terminates by outputting a message m and a signcryptext σ.

The adversary wins the game if $m = USC(pk_S, sk_R, \sigma)$ and the signcryption oracle never returned σ when queried on the message m. The advantage of $\mathcal{A}$ is defined as $Pr[\mathcal{A} \text{ wins}]$.

2.2 Tag-KEMs

In the traditional KEM-DEM framework for hybrid encryption, the KEM uses public key methods to encrypt and transmit the symmetric key used by the DEM. Formally, a KEM consists of an asymmetric key generation algorithm that outputs a private/public key-pair, an encapsulation algorithm that encrypts a random symmetric key using public-key techniques, and a decapsulation algorithm that uses the corresponding private key to decrypt said symmetric key from its encapsulation. This paradigm for building hybrid encryption schemes was extended in early 2005, when Abe *et.al.* [1] showed that one might build more efficient hybrid schemes by replacing the KEM with what they call a *tag-KEM*.

Definition 3 (Tag-KEM). *A tag-KEM* $TKEM = (Gen, Sym, Encap, Decap)$ *is defined as a tuple of four algorithms:*

– *A probabilistic key generation algorithm, Gen. It takes as input a security parameter* 1^k*, and outputs a private key sk and a public key pk. The public key contains all specific choices used by the scheme, such as choice of groups.*

- *A probabilistic symmetric key generation algorithm, Sym. It takes as input a public key pk, and outputs a symmetric key K and some internal state information ω.*
- *A probabilistic encapsulation algorithm, Encap. It takes as input the state information ω together with an arbitrary string τ, which is called a tag, and outputs an encapsulation E.*
- *A deterministic decapsulation algorithm, Decap. It takes a private key sk, an encapsulation E and a tag τ as input, and outputs a symmetric key K.*

For a tag-KEM to be sound, the decapsulation algorithm Decap must output the correct key K when run with a correctly formed encapsulation E of K, and the corresponding private key and tag.

Tag-KEMs as such may be viewed as a generalisation of regular KEMs: if the tag τ is a fixed string, the *Sym* and *Encap* algorithms together make up the encapsulation algorithm of the traditional model.

Definition 4 (DEM). *A data encapsulation mechanism $DEM = (Enc, Dec)$ is defined as a pair of algorithms:*

- *A symmetric encryption algorithm Enc, that takes a symmetric key $K \in \mathcal{K}$ and a message m as input, and returns a ciphertext $C = Enc_K(m)$. The set $\mathcal{K}$ is called the keyspace of the DEM.*
- *A symmetric decryption algorithm Dec, that takes a symmetric key $K \in \mathcal{K}$ and a ciphertext c as input, and returns a message $m = Dec_K(C)$.*

For soundness, the encryption and decryption algorithms should be each other's inverses under a fixed key K. Notationally, $m = Dec_K\big(Enc_K(m)\big)$.

For the purposes of this paper, it is only required that DEMs are secure with respect to indistinguishability against *passive* attackers (IND-PA). Formally, this security notion is captured by the following game, played between a challenger and a two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

1. The challenger generates a random symmetric $K \in \mathcal{K}$.
2. The adversary runs $\mathcal{A}_1$ with the security parameter 1^k as input. $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, as well as some state information *state*.
3. The challenger generates a random bit $b \in \{0, 1\}$ and computes the challenge ciphertext $C = Enc_K(m_b)$.
4. The adversary runs $\mathcal{A}_2$ on the input $(state, C)$. $\mathcal{A}_2$ terminates by returning a guess b' for the value of b.

The adversary wins the game whenever $b = b'$. The advantage of $\mathcal{A}$ is defined as $\big|Pr[b = b'] - 1/2\big|$.

A tag-KEM may be combined with a DEM to form a hybrid encryption scheme in a similar way as a regular KEM. However, in [1] the composition is

$Encr(pk, m)$: $(K, \omega) \overset{R}{\leftarrow} Sym(pk)$. $C \leftarrow Enc_K(m)$. $E \overset{R}{\leftarrow} Encap(\omega, C)$. $\sigma \leftarrow (E, C)$. Return σ.	$Decr(sk, \sigma)$: $(E, C) \leftarrow \sigma$. $K \leftarrow Decap(sk, E, C)$. $m \leftarrow Dec_K(C)$. Return m. $Key(1^k)$: $(sk, pk) \overset{R}{\leftarrow} Key(1^k)$. Return (sk, pk).

Fig. 1: Construction of asymmetric encryption scheme from a tag-KEM and DEM.

done in a novel manner, using the ciphertext output by the DEM as the tag. The explicit construction is shown in Figure 1.

The main result of Abe *et.al.* [1] is that the construction of Figure 1 is IND-CCA2 secure, provided that the DEM is secure against passive attackers (IND-PA), and it is not possible for an adversary, given a pair (E, K), to determine whether K is the key encapsulated by E, or a random key of the correct length. This contrasts with the traditional KEM-DEM construction, in which the DEM is required to be secure against an active attack for the resulting hybrid encryption scheme to be IND-CCA2.

3 Signcryption Tag-KEMs

3.1 Basic Definition

We define a Signcryption Tag-KEM (SCTK) by direct analogy to the previous definition of tag-KEMs for encryption.

Definition 5 (Signcryption Tag-KEM). *A signcryption tag-KEM* $SCTK = (Com, Key_S, Key_R, Sym, Encap, Decap)$ *is defined as a tuple of six algorithms.*

- *A probabilistic common parameter generation algorithm, Com. It takes as input a security parameter* 1^k*, and returns all the global information* I *needed by users of the scheme, such as choice of groups or hash functions.*
- *A probabilistic sender key generation algorithm,* Key_S*. It takes as input the global information* I*, and outputs a private/public keypair* (sk_S, pk_S) *that is used to send signcrypted messages.*
- *A probabilistic receiver key generation algorithm,* Key_R*. It takes as input the global information* I*, and outputs a private/public keypair* (sk_R, pk_R) *that is used to receive signcrypted messages.*
- *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*[3] *key encapsulation algorithm, Encap. It takes as input the state information ω and an arbitrary tag τ, and returns an encapsulation E.*
– *A deterministic decapsulation/verification algorithm, Decap. It takes as input the sender's public key pk_S, the receiver's private key sk_R, an encapsulation E and a tag τ. The algorithm returns either a symmetric key K or the unique error symbol $\perp$.*

For the SCTK to be sound, the decapsulation/verification algorithm must return the correct key K whenever the encapsulation E is correctly formed and the corresponding keys and tag are supplied.

The basic idea behind a signcryption tag-KEM is that the key encapsulation algorithm provides what amounts to a signature on the tag τ. Signcryption tag-KEMs may thus be combined with regular DEMs to form a hybrid signcryption scheme as shown in Figure 2, using the SCTK to provide a signature on the symmetric ciphertext c and encapsulate the symmetric key K.

$Com(1^k)$:
$I \xleftarrow{R} Com(1^k)$.
Return I.

$Keys_S(I)$:
$(sk_S, pk_S) \xleftarrow{R} Keys_S(I)$.
Return (sk_S, pk_S).

$Key_R(I)$:
$(sk_R, pk_R) \xleftarrow{R} Key_R(I)$.
Return (sk_R, pk_R).

$SC(sk_S, pk_R, m)$:
$(K, \omega) \xleftarrow{R} Sym(sk_S, pk_R)$.
$C \leftarrow Enc_K(m)$.
$E \xleftarrow{R} Encap(\omega, C)$.
$\sigma \leftarrow (E, C)$.
Return σ.

$USC(pk_S, sk_R, \sigma)$:
$(E, C) \leftarrow \sigma$.
If $\perp \leftarrow Decap(pk_S, sk_R, E, C)$:
Return $\perp$ and terminate.
Else $K \leftarrow Decap(pk_S, sk_R, E, C)$.
$m \leftarrow Dec_K(C)$.
Return m.

Fig. 2: Construction of hybrid signcryption scheme from SCTK and DEM.

Previous discussion of hybrid signcryption schemes have discussed efficient hybrid signcryption as a variant of the "Encrypt-and-Sign" [2] paradigm. A straightforward approach is to encrypt the message to be sent with a symmetric cipher, while combining the features of key encapsulation and digital signatures into one efficient operation [11, 12, 5]. Using signcryption tag-KEMs in the con-

[3] Theoretically, this algorithm can always be represented as a deterministic algorithm, which takes an appropriately sized random string as input. This random string is generated by the probabilistic algorithm *Sym* and passed to *Encap* as part of ω. However, if the probabilistic version of the encapsulation algorithm *Encap* is only expected-polynomial-time, then the deterministic version will have an (arbitrarily small) possibility of failing.

struction yields something more akin to a "Encrypt-then-Sign" based scheme, since the signature is made on the ciphertext "tag".

Another feature of the signcryption tag-KEM construction is that it automatically supports the sending of associated cleartext data with a message. In particular, one may submit a tag $\tau = (C, l)$ to the encapsulation algorithm, consisting of the ciphertext C as well as a label l containing any associated data that is to be bound to C by the encapsulation. Because the encapsulation acts as a signature on the input tag, the authenticity and integrity of both ciphertext and associated data is provided. The only technical requirement for using this construction is that the tag τ must be formatted in such a way that $(C, l) \leftarrow \tau$ may be parsed in a deterministic and unambiguous manner. A standard application of this feature is the common practice of "binding" the sender's and receiver's public key to any signcryption sent between them. Many signcryption schemes explicitly do this, in order to provide some degree of multi-user security. A similar effect may clearly be achieved by computing the signcryption of a combination of the message and a hash of the associated data. However, this requires either slightly greater bandwidth or a slightly reduced message space.

3.2 Security Models

For a signcryption tag-KEM to be considered secure, it must fulfill well-defined security notions with respect to confidentiality and authenticity/integrity. The tag-KEM confidentiality model used in [1] may easily adapted to the signcryption setting, and the notion of strong existential unforgeability is adapted to provide authenticity/integrity.

In the IND-CCA2 game for a signcryption tag-KEM, the adversary attempts to distinguish whether a given symmetric key is the one embedded in an encapsulation. The adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ runs in three stages, with each stage having access to oracles that fascilitate both adaptive encapsulation and decapsulation queries. For a given security parameter 1^k, this may be expressed by the following game:

1. The challenger generates a set of global parameters $I = \mathit{Com}(1^k)$, a sender keypair $(sk_S, pk_S) = \mathit{Key}_S(I)$ and a receiver keypair $(sk_R, pk_R) = \mathit{Key}_R(I)$.
2. The adversary runs $\mathcal{A}_1$ on the input (I, pk_S, pk_R). During its execution, $\mathcal{A}_1$ is given access to three oracles, corresponding to each of the algorithms *Sym*, *Encap* and *Decap*:
   - The symmetric key generation oracle does not take any input, and computes $(K, \omega) = \mathit{Sym}(sk_S, pk_R)$. It then stores the value ω (hidden from the view of the adversary, and overwriting any previously stored values), and returns the symmetric key K.
   - The key encapsulation oracle takes an arbitrary tag τ as input, and checks whether there exists a stored value ω. If there is not, it returns $\perp$ and terminates. Otherwise it erases the value from storage, and returns $\mathit{Encap}(\omega, \tau)$.

- The decapsulation/verification oracle takes an encapsulation E and a tag τ as input, and returns $Decap(pk_S, sk_R, E, \tau)$.

$\mathcal{A}_1$ terminates by returning state information *state₁*.

3. The challenger computes $(K_0, \omega) = Sym(sk_S, pk_R)$, generates a random symmetric key $K_1 \in \mathcal{K}$, and a random bit $b \in \{0, 1\}$.
4. The adversary runs $\mathcal{A}_2$ on the input $(state_1, K_b)$. During its execution, $\mathcal{A}_2$ may access the same oracles as previously. $\mathcal{A}_2$ terminates by returning state information $state_2$ and a tag τ.
5. The challenger computes a challenge encapsulation $E = Encap(\omega, \tau)$.
6. The adversary runs $\mathcal{A}_3$ on the input $(state_2, E)$. During its execution, $\mathcal{A}_3$ may access the same oracles as previously, with the restriction that (E, τ) may not be asked to the decapsulation oracle. $\mathcal{A}_3$ terminates by returning a guess b' for the value of b.

The adversary wins the game whenever $b = b'$. The advantage of $\mathcal{A}$ is defined as $\big|Pr[b = b'] - 1/2\big|$. A signcryption tag-KEM is said to be *IND-CCA2 secure* if, for any adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the IND-CCA2 game is negligible with respect to the security parameter 1^k.

It is important to notice the interaction between the symmetric key generation and encapsulation oracles. This is done to allow the adversary to perform completely adaptive encapsulations, without having access to the internal information stored in ω. The IND-CCA2 game ensures that a SCTK fulfills several necessary properties with regards to malleability and information hiding, and replaces the notions of IND-CCA2 and INP-CCA2 used by Dent [11, 12] for regular signcryption KEMs.

With respect to authenticity and integrity, an adversary should not be able to find encapsulation/tag-pairs (E, τ) such that $Decap(pk_S, sk_R, E, \tau) \neq \perp$, except by the way of oracles. Since the encapsulation algorithm should provide a signature on the tag τ, this is closely tied to forging the underlying signature scheme. An attack game corresponding to the sUF-CMA security of a SCTK may thus be specified as follows:

1. The challenger generates a set of global parameters $I = Com(1^k)$, a sender keypair $(sk_S, pk_S) = Key_S(I)$ and a receiver keypair $(sk_R, pk_R) = Key_R(I)$.
2. The adversary $\mathcal{A}$ is run on the input (I, pk_S, sk_R, pk_R). During its execution, $\mathcal{A}$ may access the symmetric key generation and encapsulation oracles as were defined in the previous game. $\mathcal{A}$ terminates by returning an encapsulation E and a tag τ.

The adversary wins the game if $\perp \neq Decap(pk_S, sk_R, E, \tau)$ and the encapsulation oracle never returned E when queried on the tag τ. The advantage of $\mathcal{A}$ is defined as $Pr[\mathcal{A}$ wins$]$. A signcryption tag-KEM is said to be *sUF-CMA secure* if, for any adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in the sUF-CMA game is negligible with respect to the security parameter 1^k.

Definition 6 (Secure signcryption tag-KEM). *A signcryption tag-KEM SCTK is said to be secure if it is IND-CCA2 and sUF-CMA secure.*

3.3 Generic Security of Hybrid Signcryption

If the SCTK+DEM construction is to be of any use, the resulting signcryption scheme must be provably secure.

Theorem 1. *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 and the DEM is IND-PA secure, then SC is IND-CCA2 secure.*

Proof. Let Game 0 be the regular IND-CCA2 game for signcryption, as specified in Section 2.1. In the following game, the hybrid signcryption procedure is altered to use a random key when generating the challenge signcryptext, rather than the real key output by *Sym*. We refer to the resulting game as Game 1:

1. The challenger generates a set of global parameters $I = Com(1^k)$, a sender keypair $(sk_S, pk_S) = Key_S(I)$ and a receiver keypair $(sk_R, pk_R) = Key_R(I)$.
2. The adversary runs $\mathcal{A}_1$ on the input (I, pk_S, pk_R). During its execution, $\mathcal{A}_1$ has access to signcryption and unsigncryption oracles. The signcryption oracle takes a message m as input, and returns $SC(sk_S, pk_R, m)$. The unsigncryption oracle takes a signcryptext σ as input, and returns $USC(pk_S, sk_R, \sigma)$. $\mathcal{A}_1$ terminates by outputting two messages (m_0, m_1) and some state information *state*.
3. The challenger computes $(K, \omega) = Sym(sk_S, pk_R)$, and generates a random key $K' \in \mathcal{K}$, as well as a random bit $b \in \{0, 1\}$. He then computes $C = Enc_{K'}(m_b)$ and $E = Encap(\omega, C)$, and sets $\sigma = (E, C)$.
4. The adversary runs $\mathcal{A}_2$ on the input $(state, \sigma)$. During its execution, $\mathcal{A}_2$ may access signcryption and unsigncryption oracles as above, with the restriction that σ may not be asked to the unsigncryption oracle. $\mathcal{A}_2$ terminates by outputting a guess b' for the bit b.

Let X_0 and X_1 be the events that $b = b'$ in Game 0 and Game 1, respectively. We will show that $\left|Pr[X_1] - Pr[X_0]\right| \leq 2\epsilon_{SCTK}$. Here, ϵ_{SCTK} is the advantage of a particular distinguisher algorithm $\mathcal{D}$ at attacking the IND-CCA2 security of the SCTK.

Figure 3 gives a complete specification of the algorithm $\mathcal{D}$. It plays the IND-CCA2 game against SCTK, using $\mathcal{A}$ as a subroutine. Oracle queries made by $\mathcal{A}$ are simulated by $\mathcal{D}$. It uses the subroutines $\mathcal{O}_{SC}$ to simulate signcryption oracle queries, and $\mathcal{O}_{USC}$ to simulate unsigncryption queries. The symmetric key generation, encapsulation and decapsulation/verification oracles accessible by $\mathcal{D}$ are referred to as $\mathcal{O}_S$, $\mathcal{O}_E$ and $\mathcal{O}_D$, respectively. We denote the execution of an algorithm $\mathcal{A}$ that takes input values $\alpha, \ldots$ and has access to oracles $\mathcal{O}, \ldots$ as $\mathcal{A}(\alpha, \ldots; \mathcal{O}, \ldots)$.

In Figure 3, the challenge encapsulation/tag (E, C) is only asked to the decapsulation oracle by $\mathcal{D}$ if the challenge ciphertext σ is asked to the unsigncryption oracle by $\mathcal{A}$. Note that $\mathcal{D}_2$ receives either the real key encapsulated by

$\mathcal{D}_1(I, pk_S, pk_R; \mathcal{O}_S, \mathcal{O}_E, \mathcal{O}_D)$:	$\mathcal{O}_{SC}(m)$:
$(m_0, m_1, s) \xleftarrow{R} \mathcal{A}_1(I, pk_S, pk_R; \mathcal{O}_{SC}, \mathcal{O}_{USC})$.	$K \xleftarrow{R} \mathcal{O}_S$.
$state_1 \leftarrow (m_0, m_1, s)$.	$C \leftarrow Enc_K(m)$.
Return $state_1$.	$E \xleftarrow{R} \mathcal{O}_E(C)$.
	$\sigma \leftarrow (E, C)$.
$\mathcal{D}_2(state_1, K; \mathcal{O}_S, \mathcal{O}_E, \mathcal{O}_D)$:	Return σ.
$b \xleftarrow{R} \{0, 1\}$.	
$C \leftarrow Enc_K(m_b)$.	$\mathcal{O}_{USC}(\sigma)$:
$state_2 \leftarrow (state_1, b, C)$.	$(E, C) \leftarrow \sigma$.
Return $(state_2, C)$.	If $\perp \leftarrow \mathcal{O}_D(E, C)$:
	Return $\perp$ and terminate.
$\mathcal{D}_3(state_2, E; \mathcal{O}_S, \mathcal{O}_E, \mathcal{O}_D)$:	Else $K \leftarrow \mathcal{O}_D(E, C)$.
$(m_0, m_1, s, b, C) \leftarrow state_2$. $\sigma \leftarrow (E, C)$.	$m \leftarrow Dec_K(C)$.
$b' \xleftarrow{R} \mathcal{A}_2(s, \sigma; \mathcal{O}_{SC}, \mathcal{O}_{USC})$.	Return m.
If $b = b'$: Return 1.	
Else: Return 0.	

Fig. 3: Distinguisher algorithm $\mathcal{D}$.

E or a random key from the challenger. If $\mathcal{D}_2$ receives the real key, then b' is the output $\mathcal{A}$ would produce when playing Game 0. Similarly, if $\mathcal{D}_2$ receives a random key, then b' is the output $\mathcal{A}$ would produce when playing Game 1. The following derivation is well known:

$$\begin{aligned}\left|Pr[\mathcal{D} \text{ wins}] - \frac{1}{2}\right| &= \frac{1}{2}\Big|Pr[\mathcal{D} \text{ outputs } 1 | \mathcal{D} \text{ received real key } K] \\ &\quad - Pr[\mathcal{D} \text{ outputs } 1 | \mathcal{D} \text{ received random key } K]\Big| \\ &= \frac{1}{2}\Big|Pr[b = b' | \mathcal{D} \text{ received real key } K] \\ &\quad - Pr[b = b' | \mathcal{D} \text{ received random key } K]\Big| \\ &= \frac{1}{2}\big|Pr[X_1] - Pr[X_0]\big|.\end{aligned}$$

Hence, the difference in $\mathcal{A}$'s advantage between Game 0 and Game 1 is bounded by twice that of an adversary against the IND-CCA2 security of SCTK.

We proceed to show that the advantage of $\mathcal{A}$ in Game 1 is bounded by that of a passive attacker against the DEM. Figure 4 specifies an adversary $\mathcal{B}$ against the IND-PA security of the DEM, that uses $\mathcal{A}$ as a subroutine. In the game described in Figure 4, $\mathcal{B}$ simulates the environment of $\mathcal{A}$ in Game 1 perfectly. Furthermore, $\mathcal{B}$ wins every time $\mathcal{A}$ would have won Game 1. Hence, they have the same advantage. It follows that

$$\epsilon_{SC} \leq 2\epsilon_{SCTK} + \epsilon_{DEM}, \tag{1}$$

where ϵ_{SC}, ϵ_{SCTK} and ϵ_{DEM} are the advantages of adversaries against IND-CCA2 security of the hybrid signcryption scheme, the IND-CCA2 security of

$\mathcal{B}_1$:	$\mathcal{O}_{SC}(m)$:
$I \xleftarrow{R} Com(1^k)$.	$(K, \omega) \xleftarrow{R} Sym(sk_S, pk_R)$.
$(sk_S, pk_S) \xleftarrow{R} Keys_S(I)$.	$C \leftarrow Enc_K(m)$.
$(sk_R, pk_R) \xleftarrow{R} Key_R(I)$.	$E \xleftarrow{R} Encap(\omega, C)$.
$(m_0, m_1, s) \xleftarrow{R} \mathcal{A}_1(I, pk_S, pk_R; \mathcal{O}_{SC}, \mathcal{O}_{USC})$.	$\sigma \leftarrow (E, C)$.
$state \leftarrow (I, sk_S, pk_S, sk_R, pk_R, m_0, m_1, s)$.	Return σ.
Return $(m_0, m_1, state')$.	
	$\mathcal{O}_{USC}(\sigma)$:
$\mathcal{B}_2(state, C)$:	$(E, C) \leftarrow \sigma$.
$(I, sk_S, pk_S, sk_R, pk_R, m_0, m_1, s) \leftarrow state$.	If $\perp \leftarrow Decap(pk_S, sk_R, E, C)$:
$(K, \omega) \xleftarrow{R} Sym(sk_S, pk_R)$.	Return $\perp$ and terminate.
$E \xleftarrow{R} Encap(\omega, C)$.	Else $K \leftarrow Decap(pk_S, sk_R, E, C)$.
$\sigma \leftarrow (C, E)$.	$m \leftarrow Dec_K(C)$.
$b \xleftarrow{R} \mathcal{A}_2(s, \sigma; \mathcal{O}_{SC}, \mathcal{O}_{USC})$.	Return m.
Return b.	

Fig. 4: IND-PA adversary against the DEM.

the signcryption tag-KEM and the IND-PA security of the DEM, respectively. □

Remark 1. This reduction is significantly tighter than those found for regular hybrid signcryption in [11, 5]. In the original approach to hybrid signcryption, the confidentiality proof relies on four terms: the indistinguishability of the symmetric keys the KEM produces, the unforgeability of the KEM, the ability of the KEM to disguise the messages and the passive security of the DEM. This is particularly inefficient as many proofs of unforgeability contain weak security reductions. We see this improved security result, and the comparative simplicity of proving the security of a signcryption tag-KEM, as the main advantages of the SCTK paradigm.

Theorem 2. *Let SC be a hybrid signcryption scheme constructed from a signcryption tag-KEM and a DEM. If the signcryption tag-KEM is sUF-CMA secure, then SC is also sUF-CMA secure.*

Proof. Since every valid forgery of SC implies a valid encapsulation, it is reasonably straightforward to show that forgery of SC implies forgery of the underlying SCTK. Figure 5 specifies an adversary $\mathcal{B}$, which uses a black-box adversary $\mathcal{A}$ against the UF-CMA security of SC to win the corresponding sUF-CMA game against SCTK. In the above scenario, $\mathcal{A}$ wins the forgery game against SC whenever the returned σ unsigncrypts to m and m has not been queried to the signcryption oracle $\mathcal{O}_{SC}$. If this is the case, then $\mathcal{B}$ wins the sUF-CMA game against SCTK.

To see this, note that $\mathcal{B}$ wins whenever it returns a pair (E, C) that does not decapsulate to $\perp$ and such that E was never a response from $\mathcal{O}_E$ to a query C. Since σ is a valid ciphertext, the former condition is always fulfilled. Furthermore,

$\mathcal{B}(I, pk_S, sk_R, pk_R; \mathcal{O}_S, \mathcal{O}_E)$:	$\mathcal{O}_{SC}(m)$:
$(m, \sigma) \xleftarrow{R} \mathcal{A}(I, pk_S, sk_R, pk_R; \mathcal{O}_{SC})$.	$K \xleftarrow{R} \mathcal{O}_S$.
$(E, C) \leftarrow \sigma$.	$C \leftarrow Enc_K(m)$.
Return (E, C).	$E \xleftarrow{R} \mathcal{O}_E(C)$.
	$\sigma \leftarrow (E, C)$.
	Return σ.

Fig. 5: Construction of a sUF-CMA adversary against SCTK.

one may note that the ciphertext σ is associated deterministically to m through the decapsulation algorithm. Hence, σ has been returned by $\mathcal{O}_{SC}$ if and only if m was ever queried. This implies that (E, C) was a query/response pair from $\mathcal{O}_{SC}$ if and only if (m, σ) was a query/response pair from $\mathcal{O}_E$. Hence, $\mathcal{B}$ wins every time $\mathcal{A}$ does.

It follows that

$$\epsilon_{SC} \leq \epsilon_{SCTK}, \tag{2}$$

where ϵ_{SC} is the advantage of the UF-CMA adversary against SC, and ϵ_{SCTK} is the advantage of the resulting sUF-CMA adversary against SCTK. $\square$

4 Sample schemes

4.1 Zheng Signcryption Revisited

Zheng's original signcryption scheme [17] has become somewhat of a canonical reference when hybrid signcryption is discussed [11, 5]. It is therefore natural to see whether it can be adapted to fit the new generic framework as well. Since Zheng's original scheme essentially uses the KEM to sign message plaintext, this requires only minor alterations.

Figure 6 gives a complete specification of a signcryption tag-KEM that, when combined with a DEM as per Figure 2, yields something very similar to Zheng's original scheme. The only difference between the schemes is that the tag τ used by *Encap* is the ciphertext $C \leftarrow Enc_K(m)$, rather than m itself. It is well-established that both Zheng's signcryption scheme and the derived signcryption KEM are secure [3, 11, 5], and it is therefore not surprising that the signcryption tag-KEM specified in Figure 6 is secure as well.

Theorem 3. *Zheng-SCTK, as specified in Figure 6, is a secure signcryption tag-KEM.*

The proof of the theorem follows those of [3, 5], and is reproduced in full in Appendix A. The specific bound obtained is similar to that of the original scheme-specific reduction [3]. If we compare this to the results of Bjørstad [5], we find that the security bound in Dent's model for hybrid signcryption [**?**] based on regular KEMs includes an additional term stemming from the unforgeability of the scheme. This illustrates the usefulness of our new construction, as it provides significantly tighter generic security bounds that correspond better with reality.

$Com(1^k)$:
Pick a k-bit prime p.
Pick a large prime q that divides $p-1$.
Pick $g \in \mathbb{Z}_q^*$ of order q.
Pick cryptographic hash functions:
$\mathcal{G} : \{0,1\}^* \to \mathcal{K}$.
$\mathcal{H} : \{0,1\}^* \to \mathbb{Z}/q\mathbb{Z}$.
$I \leftarrow (p, q, g, \mathcal{G}, \mathcal{H})$.
Return I.

$Key_S(I)$:
$sk_S \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$pk_S \leftarrow g^{sk_S} \mod p$.
Return (sk_S, pk_S).

$Key_R(I)$:
$sk_R \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$pk_R \leftarrow g^{sk_R} \mod p$.
Return (sk_R, pk_R).

$Sym(sk_S, pk_R)$:
$n \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$\kappa \leftarrow {pk_R}^n \mod p$.
$bind \leftarrow pk_S || pk_R$.
$K \leftarrow \mathcal{G}(\kappa)$.
$\omega \leftarrow (sk_S, n, \kappa, bind)$.
Return (K, ω).

$Encap(\omega, \tau)$:
$(sk_S, n, \kappa, bind) \leftarrow \omega$.
$r \leftarrow \mathcal{H}(\tau || bind || \kappa)$.
$s \leftarrow n/(sk_S + r) \mod q$.
$E \leftarrow (r, s)$.
Return E.

$Decap(pk_S, sk_R, E, \tau)$:
$(r, s) \leftarrow E$.
$\kappa \leftarrow (pk_S \cdot g^r)^{s \cdot sk_R} \mod p$.
$r' \leftarrow \mathcal{H}(\tau || bind || \kappa)$.
If $r \neq r'$:
Return $\perp$ and terminate.
Else $K \leftarrow \mathcal{G}(\kappa)$.
Return K.

Fig. 6: Zheng-SCTK.

Other existing signcryption schemes may also be representable as signcryption tag-KEMs. For example, it appears likely that the hybrid signcryption scheme of Malone-Lee [15] could also be adapted to the signcryption tag-KEM paradigm, along with its corresponding proof of security.

4.2 The CM signcryption tag-KEM

As discussed in [17, 5], the Zheng signcryption scheme is constructed by modifying an existing signature scheme. By making the randomiser κ computed during signature verification dependent on the receiver's key sk_S, an efficient signcryption scheme is constructed at a very low additional cost. This trick may be applied to other signature schemes as well. In this section, we propose a new signcryption tag-KEM, built from a recent signature scheme due to Chevallier-Mames [8]. The resulting construction has tight security reductions with respect to the Computational Diffie-Hellman and Gap Diffie-Hellman problems. This is of practical interest, since previous hybrid signcryption schemes have had relatively loose security reductions with respect to unforgeability. Figure 7 gives a complete specification of the CM signcryption tag-KEM.

$Com(1^k)$: Pick a large prime q. Let G be a cyclic group of order q, such that the representation of the elements of G is included in $\{0,1\}^k$. Pick a generator g of G. Pick cryptographic hash functions: $\mathcal{G} : G \to G$. $\mathcal{H} : \{0,1\}^* \times G^6 \to \mathbb{Z}_q$. $KDF : G \to \mathcal{K}$. $I \leftarrow (q, G, g, \mathcal{G}, \mathcal{H}, KDF)$. Return I. $Key_S(I)$: $sk_S \xleftarrow{R} \mathbb{Z}_q$. $pk_S \leftarrow g^{sk_S}$. Return (sk_S, pk_S). $Key_R(I)$: $sk_R \xleftarrow{R} \mathbb{Z}_q$. $pk_R \leftarrow g^{sk_R}$. Return (sk_R, pk_R).	$Sym(sk_S, pk_R)$: $n \xleftarrow{R} \mathbb{Z}_q$. $u \leftarrow {pk_R}^n$. $K \leftarrow KDF(u)$. $\omega \leftarrow (sk_S, pk_R, n, u)$. Return (K, ω). $Encap(\omega, \tau)$: $(sk_S, pk_R, n, u) \leftarrow \omega$. $h \leftarrow \mathcal{H}(u)$. $z \leftarrow h^{sk_S}$. $v \leftarrow h^n$. $c \leftarrow \mathcal{G}(\tau \| pk_R, pk_S, g, z, h, u, v)$. $s \leftarrow n + c \cdot sk_S, \mod q$. $E \leftarrow (z, c, s)$. $Decap(pk_S, sk_R, E, \tau)$: $u \leftarrow (g^s \cdot {pk_S}^{-c})^{sk_R}$. $h \leftarrow \mathcal{H}(u)$. $v \leftarrow h^s \cdot z^{-c}$. If $c \neq \mathcal{G}(\tau \| pk_R, pk_S, g, z, h, u, v)$: Return $\perp$. Else $K \leftarrow KDF(u)$. Return K.

Fig. 7: The CM signcryption tag-KEM

Theorem 4. *The CM signcryption tag-KEM specified in Figure 7 is a secure signcryption tag-KEM.*

A full proof is given in Appendix B. The proof uses techniques that are directly analogous to those used in the security proofs for Zheng's scheme [3, 5]. However, this scheme has a better security reduction for authenticity/integrity, since the security of the underlying signature scheme does not rely on a "forking lemma" argument [16]. To the authors' knowledge, this construction gives us the tightest known security reductions for a signcryption scheme.

As a side note, we remark that, in order to prove the integrity/authenticity of the CM signcryption tag-KEM, it was necessary to prove that the Chevallier-Mames signature scheme was strongly unforgeable. A proof of this fact was developed independently by Chevallier-Mames[9].

4.3 Signcryption schemes with associated data

Given a secure signcryption scheme with support for associated plaintext data, there exists a general construction of a secure signcryption tag-KEM. This is useful whenever the original signcryption scheme carries restrictions on the size of its message space. The implied signcryption scheme (constructed from SCTK+DEM) is identical to the construction given previously by Dodis *et.al.* [14][Theorem 4].

Let $\mathcal{SC}$ be a signcryption scheme with support for associated data, i.e. where the signcryption and unsigncryption algorithms take input of the form $m^* = (m, l)$ and $\sigma^* = (\sigma, l)$ respectively, and can parse these strings deterministically and unambigiously[4]. The resulting signcryption tag-KEM uses the common parameter and private/public key generation algorithms specified by $\mathcal{SC}$, and constructs algorithms for symmetric key generation, encapsulation and decapsulation as shown in Figure 8.

$\mathit{Sym}(\mathit{sk}_S, \mathit{pk}_R)$: $K \xleftarrow{R} \mathcal{K}$. $\omega \leftarrow (\mathit{sk}_S, \mathit{pk}_R, K)$. Return (ω, K).	$\mathit{Encap}(\omega, \tau)$: $(\mathit{sk}_S, \mathit{pk}_R, K) \leftarrow \omega$. Return $\mathit{SC}\big(\mathit{sk}_S, \mathit{pk}_R, (K, \tau)\big)$. $\mathit{Decap}(\mathit{pk}_S, \mathit{sk}_R, E, \tau)$: Return $\mathit{USC}\big(\mathit{pk}_S, \mathit{sk}_R, (E, \tau)\big)$.

Fig. 8: Constructing a SCTK from a signcryption scheme with associated data.

Theorem 5. *The signcryption tag-KEM specified in Figure 8 is as secure as the underlying signcryption scheme $\mathcal{SC}$.*

[4] The security notions for confidentiality and authenticity/integrity are modified accordingly.

This result is established through the explicit construction of algorithms that use adversaries against the derived signcryption tag-KEM to attack the underlying signcryption scheme. Intuitively, a successful IND-CCA2 adversary against the signcryption tag-KEM must distinguish between two random keys signcrypted by $\mathcal{SC}$, whereas a successful sUF-CMA adversary against the signcryption tag-KEM must create a sUF-CMA forgery of $\mathcal{SC}$. A full proof is given in Appendix C.

5 Building Better Key Agreement Mechanisms with Signcryption Tag-KEMs

The idea that signcryption KEMs can be used as key agreement mechanisms was first investigated by Dent [13]. Dent notes that whilst an encryption KEM provides a basic mechanism for agreeing a symmetric key between two parties, it does not provide any form of authentication or freshness guarantee. Moreover, he notes that signcryption KEMs (with outsider security) can be used to agree a symmetric key with authentication. A simple protocol key agreement protocol is then proposed, wherein freshness is guaranteed by the computing the MAC of a timestamp or nonce using the newly agreed symmetric key. However, as the paper remarks, this protocol is susceptible to a known key attack and should not be used in practice.

In this section we propose that signcryption tag-KEMs can be used as practical key agreement mechanisms, with the SCTK providing both the authentication and freshness components of the protocol in a simple way. Consider the following protocol which allows Alice and Bob to agree a key for a session with an ID *SID* between them:

1. Alice generates a random nonce r_A of an agreed length, and sends r_A to Bob.
2. Bob computes $(K, \omega) = \mathit{Sym}(sk_{Bob}, pk_{Alice})$ and $E = \mathit{Encap}(\omega, \tau)$ using the (unique) tag $\tau = r_A || \mathit{SID}$. Bob accepts K as the shared secret key, and sends C to Alice.
3. Alice computes $K = \mathit{Decap}(pk_{Bob}, sk_{Alice}, E, \tau)$ using the tag $\tau = r_A || \mathit{SID}$, and accepts K as the shared key providing $K \neq \perp$.

We argue that this protocol has the following attributes:

- **Implicit key authentication to both parties**. If both parties obtain the other's correct public key, then no attacker can distinguish between a session's correct public key and a randomly generated key without breaking the confidentiality criterion for the SCTK.
- **Resistance to known key attacks**. It is easy to see that an attacker that gains a key from any earlier protocol execution (or, indeed, in a later protocol execution) between Alice and Bob gains no advantage in breaking the scheme. This is because this "session corruption" is equivalent to making a signcryption oracle query with a random tag. Since the SCTK remains secure in this situation, so does the key agreement protocol.

– **Key confirmation from Bob to Alice**. Since no party (including Alice) can forge a signcryptext that purports to come from Bob, if Alice recovers a key K from C, then that key K *must* have been produced by Bob in the correct way. Therefore, Alice can have confidence that Bob knows the correct key. However, an extra round of interaction will be required if Alice wishes to give Bob key confirmation.

We argue that this derivation is useful because it finally gives a secure way to use KEMs for key establishment. Of course, a secure signcryption scheme can always be used as a key transport mechanism; however, it was not previously known if signcryption-style techniques could be used for key agreement. The aforementioned protocol settles this question. Whether an individual signcryption tag-KEM should be regarded as a key transport or key agreement mechanism depends upon its individual characteristics.

Unfortunately, we cannot provide a formal proof of the security of this protocol in any of the standard models [4, 6, 7]. This is not due to any property of this protocol, and a proof of security in the Bellare-Rogaway model [4, 6] intuitively seems fairly simple. However, such a proof requires the use of an SCTK that is secure in a multi-party model. While it appears straightforward to extend the SCTK framework to a multi-party setting, it may be more complicated to adapt the security proofs for specific schemes. Nevertheless, we suggest that the key agreement protocol derived from any SCTK given in Section 4 are secure, efficient and useable.

6 Conclusions

We have shown that there is a natural extension of the concept of a tag-KEM to the signcryption setting and proven that secure signcryption tag-KEMs can be combined with passively secure DEMs to provide signcryption schemes with full insider security. This vastly simplifies and improves upon the KEM-DEM model insider secure signcryption schemes proposed by Dent [12]. To show that this construction is viable, we have given several examples of signcryption Tag-KEMs, including a brand new construction based on the Chevallier-Mames signature scheme with very tight security bounds.

Acknowledgements

Tor E. Bjørstad wishes to thank the ECRYPT project and the Norwegian Research Council for their generous financial support. Alexander W. Dent wishes to think the ECRYPT project and the EPSRC's Junior Research Fellowship programme for their generous financial support. Both authors wish to thank the PKC 2006 anonymous reviewers for their comments.

References

1. Asayuki Abe, Rosario Gennaro, Kaoru Kurosawa, and Victor Shoup. Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of

Kurosawa-Desmedt KEM. In *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 128–146. Springer–Verlag, 2005.
2. Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer–Verlag, 2002.
3. Joonsang Baek, Ron Steinfeld, and Yuliang Zheng. Formal proofs for the security of signcryption. In *Proceedings of PKC 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 80–98. Springer–Verlag, 2002.
4. M. Bellare and P. Rogaway. Entity authentication and key distribution. In D. R. Stinson, editor, *Advances in Cryptology – Crypto '93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 1993.
5. Tor E. Bjørstad. Provable security of signcryption. Master's thesis, Norwegian University of Technology and Science, 2005. http://www.nwo.no/~tor/pdf/msc_thesis.pdf.
6. S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In M. Darnell, editor, *Cryptography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer-Verlag, 1997.
7. R. Canetti and H. Krawcyzk. Universally composable notions of key exchange and secure channels. In L. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351. Springer-Verlag, 2002.
8. Benoît Chevallier-Mames. An efficient CDH-based signature scheme with a tight security reduction. In *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 511–526. Springer–Verlag, 2005.
9. Benoît Chevallier-Mames. Personal correspondence, 2005.
10. Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2004.
11. Alexander W. Dent. Hybrid cryptography. Cryptology ePrint Archive, Report 2004/210, 2004. http://eprint.iacr.org/2004/210/.
12. Alexander W. Dent. Hybrid signcryption schemes with insider security. In *Proceedings of ACISP 2005*, volume 3574 of *Lecture Notes in Computer Science*, pages 253–266. Springer–Verlag, 2005.
13. Alexander W. Dent. Hybrid signcryption schemes with outsider security. In *Proceedings of ISC 2005*, volume 3650 of *Lecture Notes in Computer Science*, pages 203–217. Springer–Verlag, 2005.
14. Yevgeniy Dodis, Michael J. Freedman, Stanislaw Jarecki, and Shabsi Walfish. Optimal signcryption from any trapdoor permutation. Cryptology ePrint Archive, Report 2004/020, 2004. http://eprint.iacr.org/2004/020/.
15. John Malone-Lee. Signcryption with non-interactive non-repudiation. Technical Report CSTR-02-004, Department of Computer Science, University of Bristol, 2004. http://www.cs.bris.ac.uk/Publications/Papers/1000628.pdf.
16. David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070, pages 387–398. Springer–Verlag, 1996.
17. Yuliang Zheng. Digital signcryption or how to achieve cost (signature & encryption) << cost (signature) + cost (encryption). In *Advances in Cryptology – CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer–Verlag, 1997. Unpublished full ver-

sion (47 pages), dated 1999, available through the author's home page http://www.sis.uncc.edu/~yzheng/papers/signcrypt.pdf.

A Proof of Theorem 3

Zheng's signcryption tag-KEM is a direct adaptation of the original signcryption scheme, which is known to be secure [17, 3]. Proving the SCTK secure is therefore little more than applying the previous proof techniques in the new generic setting.

A.1 sUF-CMA security of Zheng-SCTK

The signature scheme underlying Zheng's signcryption scheme is known as SDSS1 [17]. Our proof of security for Zheng-SCTK is in the form of a reduction to existential forgery of SDSS1-signatures, which may be shown to be sUF-CMA secure with respect to the discrete logarithm problem using "standard" forking lemma techniques [16]. A concrete description of the scheme is given in Figure 9. A sUF-

$Key(1^k)$: Pick a k-bit prime p. Pick a large prime q that divides $p-1$. Pick $g \in \mathbb{Z}_q^*$ of order q. Pick a cryptographic hash function H : $\{0,1\}^* \rightarrow \mathbb{Z}/q\mathbb{Z}$. $x \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$. $y \leftarrow g^x \mod p$. $sk \leftarrow (p, q, g, \mathcal{H}, x, y)$. $pk \leftarrow (p, q, g, \mathcal{H}, y)$. Return (sk, pk).	$Sign(sk, m)$: $n \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$. $\kappa \leftarrow g^n \mod p$. $r \leftarrow H(m\|\|\kappa)$. $s \leftarrow n/(x+r) \mod q$. $\sigma \leftarrow (r, s)$. Return σ. $Ver(pk, m, \sigma)$: $\kappa \leftarrow (y \cdot g^r)^s \mod p$. $r' \leftarrow \mathcal{H}(m\|\|\kappa)$. If $r = r'$, return $\top$. Else return $\bot$.

Fig. 9: The SDSS1 signature scheme.

CMA forger of the SDSS1 scheme thus needs to obtain (m, κ, r, s) such that the cyclical relation

$$\begin{aligned} s &\leftarrow n/(x+r) \mod q \\ \kappa &\leftarrow (y \cdot g^r)^s \mod p \\ r &\leftarrow \mathcal{H}(m||\kappa) \end{aligned} \tag{3}$$

holds for the specified public (and the corresponding private) key. Figure 10 gives a concrete specification of such a forger $\mathcal{F}$, that accomplishes this by using a successful sUF-CMA adversary $\mathcal{A}$ against Zheng-SCTK to obtain said relation.

During the simulation, the forger may access to a signature oracle $\mathcal{O}_{Sign}$ that produces valid signatures σ on arbitrary messages m, as well as a random oracle $\mathcal{O}_H$ representing the cryptographic hash function H. These are used to simulate the symmetric key generation and encapsulation oracles used by $\mathcal{A}$. $\mathcal{F}$ also runs rigged versions of the random oracles $\mathcal{G}$ and $\mathcal{H}$. From the way the cryptographic

$\mathcal{F}(pk; \mathcal{O}_{Sign}, \mathcal{O}_H)$:
Form I and pk_S from pk.
$(sk_R, pk_R) \overset{R}{\leftarrow} Key_R(I)$.
$bind \leftarrow pk_S || pk_R$.
$(E, \tau) \overset{R}{\leftarrow} \mathcal{A}(I, pk_S, sk_R, pk_R; \mathcal{O}_S, \mathcal{O}_E, \mathcal{G}_{sim}, \mathcal{H}_{sim})$.
Return the signature $E = (r, s)$ on the message $\tau || bind$.

$\mathcal{O}_S$:
$K \overset{R}{\leftarrow} \mathcal{K}$.
Store K, overwriting any previous value.
Return K.

$\mathcal{O}_E(\tau)$:
If no stored K exists, return $\perp$.
Else read K and erase it from storage.
$(r, s) \overset{R}{\leftarrow} \mathcal{O}_{Sign}(\tau || bind)$.
$\kappa_{SDSS1} \leftarrow (pk_S \cdot g^r)^s \mod p$.
$\kappa \leftarrow \kappa_{SDSS1}^{sk_R} \mod p$.
Add (κ, K) to the I/O list of $\mathcal{G}_{sim}$.
Add $(\tau || bind || \kappa, r)$ to the I/O list of $\mathcal{H}_{sim}$.
Return (r, s).

$\mathcal{G}_{sim}(\kappa)$:
If (κ, K) is in the I/O list of $\mathcal{G}_{sim}$:
Return K and terminate.
$K \overset{R}{\leftarrow} \mathcal{K}$.
Add (κ, K) to the I/O list of $\mathcal{G}_{sim}$.
Return K.

$\mathcal{H}_{sim}(\tau || bind || \kappa)$:
If $(\tau || bind || \kappa, r)$ is in the I/O list of $\mathcal{H}_{sim}$:
Return r and terminate.
$\kappa_{SDSS1} \leftarrow \kappa^{\frac{1}{sk_R}} \mod p$.
$r \leftarrow \mathcal{O}_H(\tau || bind || \kappa_{SDSS1})$.
Add $(\tau || bind || \kappa, r)$ to the I/O list of $\mathcal{H}_{sim}$.
Return r.

Fig. 10: Forgery algorithm $\mathcal{F}$.

hash function $\mathcal{H}$ is simulated by $\mathcal{H}_{sim}$, the SCTK-adversary $\mathcal{A}$ outputs pairs (E, τ) that are precisely of the form desired by the forger $\mathcal{F}$. With regards to the strong unforgeability criterion, note that $\mathcal{A}$ only wins the sUF-CMA game against SCTK if $\mathcal{O}_E$ never returned the encapsulation E when queried on the tag τ. However, since the encapsulation oracle is created by running $\mathcal{O}_{Sign}$ on τ and only modifying the behaviour of the random oracles, this implies that $\mathcal{O}_{Sign}$ never returned (r, s) as a signature on τ either. To show that Zheng-SCTK is sUF-CMA secure with respect to the sUF-CMA security of SDSS1 signatures, it is hence only necessary to bound the probability of simulation failure.

The initial input values given to $\mathcal{A}$ are clearly of the correct form and distribution, since pk_S and I are derived from the SDSS1 public key pk, while sk_R and pk_R are output by Key_R as usual. Furthermore, the symmetric key generation oracle $\mathcal{O}_S$ and encapsulation oracles $\mathcal{O}_E$ returns values that have the correct distributions (thanks to $\mathcal{O}_{Sign}$) and are consistent with subsequent queries to the random oracle simulators. Looking at Figure 10, one may note that the encapsulation oracle returns the same $E = (r, s)$ as that which was returned by $\mathcal{O}_{Sign}$, while computing the "correct" value of κ by computing κ_{SDSS1} and computing $\kappa_{SDSS1}^{sk_R} \mod p$. The logical counterpart to this operation occurs when $\mathcal{H}_{sim}$ performs a reverse exponentiation by $\frac{1}{sk_R}$ before querying $\mathcal{O}_H$. With this detail out of the way, the lazy evaluation performed by $\mathcal{G}_{sim}$ and $\mathcal{H}_{sim}$ is perfectly legitimate, and produces consistent results.

The only possible cause of error in $\mathcal{F}$'s simulation hence occurs if $\mathcal{O}_E$ causes an inconsistency when modifying either of the I/O lists. This will happen since the values of κ, K and r are being forced by $\mathcal{O}_S$ and $\mathcal{O}_{Sign}$, whereas previous queries to $\mathcal{G}_{sim}$ and $\mathcal{H}_{sim}$ may already have assigned them.

An absolute worst case scenario occurs when $\mathcal{A}$ asks q_G different values of κ to $\mathcal{G}_{sim}$, followed by q_H queries of $(\tau||bind||\cdot)$ to $\mathcal{H}_{sim}$ with q_H new values of κ different from each other as well as the first. In this case, $(q_G + q_H)$ values of κ will be reserved by at least one oracle simulator. Each subsequent query to $\mathcal{O}_E$ will have to miss these reserved values, and will fix a new value of κ as well. The probability of the i'th query creating an inconsistency is hence at most $\frac{q_G+q_H+(i-1)}{q}$. Summing over q_E such queries yields a total failure probability of at most

$$\frac{q_E(2q_G + 2q_H + q_E - 1)}{2q}. \tag{4}$$

This probability is negligible in q, and the advantage of $\mathcal{F}$ at forging SDSS1 signatures is hence negligibly close to that of $\mathcal{A}$ at forging Zheng-SCTK. □

A.2 IND-CCA2 security of Zheng-SCTK

The main intuition behind a confidentiality proof for Zheng-SCTK is that an adversary has no great advantage in the IND-CCA2 game unless he can determine the value of κ corresponding to the challenge encapsulation. This is explicitly true when we model the cryptographic hash function $\mathcal{G}$ as a random oracle: the adversary will have no advantage whatsoever at distinguishing the output of the random oracle from an element drawn at random from the oracle's output space, unless it actually queried the oracle on κ. It is evident that in this situation, the adversary's advantage at winning the IND-CCA2 game is bounded by the probability that they compute the value of κ. For all intents and purposes, the following proof of the IND-CCA2 security of Zheng-SCTK is that of [3], translated into the signcryption tag-KEM setting.

The important value κ is, per the specification of Zheng's scheme, determined by the relation

$$\kappa \equiv {pk_R}^n \equiv ({pk_R}^{sk_S+r})^s \equiv (pk_S \cdot g^r)^{s \cdot sk_R} \mod p. \tag{5}$$

Given the public keys pk_S and pk_R, a challenge encapsulation $E = (r, s)$, as well as adaptive oracle access for encapsulation and decapsulation, how difficult is it for the IND-CCA2 adversary $\mathcal{A}$ to compute κ? As is shown in [3], an adversary who is able to do this may be tricked into solving arbitrary instances of the Gap Diffie-Hellman problem. Given a specific GDH problem instance $X(= g^x$

mod p), $Y(= g^y \mod p)$, one may pick the values r and s that form the challenge encapsulation at random, and set $pk_S \leftarrow (X \cdot g^{-rs})^{\frac{1}{s}} \mod p$ and $pk_R \leftarrow Y$. This maintains the correct distributions of all four variables, while causing κ to equal $g^{xy} \mod p$, the desired solution.

In order to present a consistent view to the adversary $\mathcal{A}$, it is necessary to simulate the oracles for symmetric key generation ($\mathcal{O}_S$), encapsulation ($\mathcal{O}_E$) and decapsulation ($\mathcal{O}_D$). One may then use the provided Decisional Diffie-Hellman (which takes three group elements as input, and returns $\top$ if and only if they are a DDH triplet, otherwise $\bot$) oracle to test all queries made by $\mathcal{A}$ to the random oracle simulators $\mathcal{O}_G$ and $\mathcal{O}_H$ for the desired κ-value. However, since the oracles presented to the IND-CCA2 adversary against Zheng-SCTK are being simulated with imperfect information, it is necessary to ensure that the probability of $\mathcal{A}$ asking κ to an oracle is kept negligibly close to that of the original (honest) game.

Figures 11-13 give a complete specification of an algorithm that solves the Gap Diffie-Hellman problem, using an efficient adversary against Zheng-SCTK as a subroutine. In the specification, the GDH solution $\kappa^* = g^{xy} \mod p$ is not known explicitly, although the values $K = \mathcal{O}_G(\kappa^*)$ and $r = \mathcal{O}_H(\tau||bind||\kappa^*)$ are used implicitly. Four lists, L_1^G, L_1^H, L_2^G and L_2^H, are used to maintain state for the random oracle simulators $\mathcal{O}_G$ and $\mathcal{O}_H$. The lists L_1^* maintain known input/output pairs for the random oracles simulators. In contrast, the lists L_2^* are used to maintain consistency in situations where the input values are only implicitly known due to incomplete information.

GDH-solver($I, X, Y; \mathcal{O}_{DDH}$):
$r \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$s \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$E \leftarrow (r, s)$.
$pk_S \leftarrow (X \cdot g^{-rs})^{\frac{1}{s}} \mod p$.
$pk_R \leftarrow Y$.
$K \xleftarrow{R} \mathcal{K}$.
$bind \leftarrow pk_S||pk_R$.

$state_1 \xleftarrow{R} \mathcal{A}_1(I, pk_S, pk_R; \mathcal{O}_S, \mathcal{O}_E, \mathcal{O}_D, \mathcal{O}_G, \mathcal{O}_H)$.
$(state_2, \tau) \xleftarrow{R} \mathcal{A}_2(state_1, K; \mathcal{O}_S, \mathcal{O}_E, \mathcal{O}_D, \mathcal{O}_G, \mathcal{O}_H)$.
$b' \xleftarrow{R} \mathcal{A}_3(state_2, E; \mathcal{O}_S, \mathcal{O}_E, \mathcal{O}_D, \mathcal{O}_G, \mathcal{O}_H)$.
If the GDH solution has not been found, return a random group element.

Fig. 11: Specification of GDH-solving algorithm.

As argued previously, all initial data sent to $\mathcal{A}$ is of the right form and has the correct distribution. It remains to show that the oracle simulators do not introduce a non-negligible probability of causing different behaviour than their real counterparts. One may begin by noting that the random oracle simulators $\mathcal{O}_G$ and $\mathcal{O}_H$ behave like regular state-based random oracles, until the value κ^*

$\mathcal{O}_G(\kappa)$:
If $\top \leftarrow \mathcal{O}_{DDH}(X, Y, \kappa)$:
Found the GDH solution!
Else if (κ, K) is in L_1^G:
Return K.
Else if $\top \leftarrow \mathcal{O}_{DDH}(Y, \phi, \kappa)$
for some (ϕ, K) in L_2^G:
Return K.
Else:
$K \xleftarrow{R} \mathcal{K}$.
Append (κ, K) to L_1^G.
Return K.

$\mathcal{O}_H(\tau||bind||\kappa)$:
If $\top \leftarrow \mathcal{O}_{DDH}(X, Y, \kappa)$:
Found the GDH solution!
Else if $(\tau||bind||\kappa, r)$ is in L_1^H:
Return r.
Else if $(\phi, \tau||bind, r)$ is in L_2^H
and $\top \leftarrow \mathcal{O}_{DDH}(Y, \phi, \kappa)$:
Return r.
Else:
$r \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
Append $(\tau||bind||\kappa, r)$ to L_1^H.
Return r.

Fig. 12: Specification of random oracle simulators.

$\mathcal{O}_S$:
$r \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$s \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
$\phi \leftarrow (pk_S \cdot g^r)^s \mod p$.
$\omega \leftarrow (r, s, \phi)$.
Store ω, overwriting any previous value.
If $\top \leftarrow \mathcal{O}_{DDH}(Y, \phi, \kappa)$ for some (κ, K) in L_1^G: Return K.
Else if (ϕ, K) is in L_2^G: Return K.
Else:
$K \xleftarrow{R} \mathcal{K}$.
Append (ϕ, K) to L_2^G.
Return K.

$\mathcal{O}_E(\tau)$:
If no stored ω exists, return $\perp$.
Else read ω from storage and erase it.
$(r, s, \phi) \leftarrow \omega$.
Append $(\phi, \tau||bind, r)$ to L_2^H.
$E \leftarrow (r, s)$.
Return E.

$\mathcal{O}_D(E, \tau)$:
$(r, s) \leftarrow E$.
$\phi \leftarrow (pk_S \cdot g^r)^s \mod p$.

If $\phi = X$: Return $\perp$ and terminate.

If there exists $(\tau'||bind||\kappa, r')$ in L_1^H such that $\tau = \tau'$ and $\top \leftarrow \mathcal{O}_{DDH}(y_r, \phi, \kappa)$:
If $r \neq r'$, return $\perp$ and terminate.
Else if there exists $(\phi', \tau'||bind, r')$ in L_2^H such that $\phi = \phi'$ and $\tau = \tau'$:
If $r \neq r'$, return $\perp$ and terminate.
Else:
$r' \xleftarrow{R} \mathbb{Z}/q\mathbb{Z}$.
Append $(\phi, \tau||bind, r')$ to L_2^H.
If $r \neq r'$, return $\perp$ and terminate.

If there exists (κ, K) in L_1^H such that $\top \leftarrow \mathcal{O}_{DDH}(pk_R, \phi, \kappa)$:
Return K.
Else if there exists (ϕ', K) in L_2^H such that $\phi = \phi'$:
Return K.
Else:
$K \xleftarrow{R} \mathcal{K}$.
Append (ϕ, K) to L_2^G.
Return K.

Fig. 13: Specification of symmetric key generation, encapsulation and decapsulation oracles.

is detected using the DDH oracle. Furthermore, the symmetric key generation oracle $\mathcal{O}_S$ chooses random values for r and s and checks L_1^G and L_2^G for previous entries before picking a new K, thus acting like its honest counterpart while avoiding any conflicts with $\mathcal{O}_G$.

The encapsulation oracle $\mathcal{O}_E$ may however cause inconsistent behaviour when it modifies L_2^H. This is because it creates the implicit relation $r = \mathcal{O}_H(\tau || bind || \phi^{sk_R} \bmod p)$. Since r is chosen before τ is specified, the oracle has no way to guarantee consistency with respect to previous entries in L_1^H and L_2^H made by $\mathcal{O}_H$, $\mathcal{O}_E$ or itself.

Consider an adversary who asks at most q_H, q_E and q_D queries to the respective oracles. In a worst case scenario for the simulation of $\mathcal{O}_E$, there may be be at most $q_H + q_D$ entries in the lists L_1^H and L_2^H when it is run the first time. Each subsequent execution of the oracle adds another entry to L_2^H. Summing the probability of failure over q_e oracle queries thus gives a total probability of $\frac{q_E(q_E+2q_D+2q_H-1)}{2q}$ that an error occurs with respect to the consistency of $\mathcal{O}_H$ in the simulation of $\mathcal{O}_E$. This is negligible with respect to q.

In the case of the final oracle, all four I/O lists are carefully checked before any entry is added, so there is no consistency problems caused during the execution. However, if the computed value ϕ is equal to X, the simulator always returns $\perp$. This is because it corresponds to κ^* being part of the query, and proceeding honestly would reveal information about $\mathcal{O}_G$ and $\mathcal{O}_H$ to $\mathcal{A}$, without learning the value of κ^*. Hence it is necessary to bound the probability that $\mathcal{O}_D$ is queried with a pair $\big(E = (r, s), \tau\big)$ that is a valid encapsulation *and* that $(pk_S \cdot g^r)^s = X$ *and* that it is not the challenge encapsulation/tag pair (E^*, τ^*).

Consider a hypothetical query $(E = (r', s'), \tau')$ for which this indeed is the case. If this is a valid encapsulation, then $\mathcal{H}(\tau', bind, \kappa^*) = r'$. Assume for the sake of contradiction that $\tau' = \tau$. Then $r' = r^*$, since the input to the random oracle is the same as for the challenge encapsulation. Furthermore, from the relation $(y_s \cdot g^{r^*})^{s^*} \equiv X \equiv (y_s \cdot g^{r'})^{s'} (\bmod p)$ and the fact that all elements apart from the identity element in $\mathbb{Z}/q\mathbb{Z}$ are of order q, we may safely conclude that $s' = s^*$ (unless the Gap Diffie-Hellman problem instance in question is trivial). Hence, assume instead that $\tau' \neq \tau^*$. In this case, the probability of $\mathcal{H}(\tau', bind, \kappa^*)$ being equal to r' is precisely $\frac{1}{q}$, since $\mathcal{H}$ is a random oracle. The probability of this occuring within q_D oracle queries is hence at most $\frac{q_D}{q}$, which is negligible with respect to q.

Adding the different failure probabilities from oracle simulation yields a total failure probability of at most

$$\frac{q_E(q_E + q_D + 2q_H - 1) + 2q_D}{2q}, \tag{6}$$

which is negligible in q, and the advantage of the GDH-solver is hence negligibly close to that of $\mathcal{A}$ winning the IND-CCA2 game against Zheng-SCTK. $\square$

B Proof of Theorem 4

The Chevallier-Mames signcryption tag-KEM is derived from the corresponding signature scheme [8]. To prove the signcryption tag-KEM secure, it is important to keep some key features of the signature scheme in mind. First of all, since the signatures themselves are secure, it is possible to exploit the relation when proving the sUF-CMA security of the SCTK. Moreover, since the signature scheme is based on a zero-knowledge protocol, it is straightforward to simulate signatures, and by extension encapsulation oracle queries, in the random oracle model. The security proofs for CM-SCTK resemble those of Zheng's scheme, which is natural given the underlying similarities.

B.1 sUF-CMA security of CM-SCTK

To prove that CM-SCTK is sUF-CMA secure, we provide a reduction to the sUF-CMA security of Chevallier-Mames signatures. A specification of the Chevallier-Mames signature scheme is given in Figure 14. Chevallier-Mames signatures are

$Com(1^k)$: Pick a large prime q. Let G be a cyclic group of order q, such that the representation of the elements of G is included in $\{0,1\}^k$. Pick a generator g of G. Pick cryptographic hash functions: $\mathcal{G} : G \to G$. $\mathcal{H} : \{0,1\}^* \times G^6 \to \mathbb{Z}_q$. $I \leftarrow (q, G, g, \mathcal{G}, \mathcal{H})$. Return I. $Key(I)$: $sk \xleftarrow{R} \mathbb{Z}_q$. $pk \leftarrow g^{sk}$. Return (sk, pk).	$Sign(sk, m)$: $n \xleftarrow{R} \mathbb{Z}_q$. $u \leftarrow g^n$. $h \leftarrow \mathcal{H}(u)$. $z \leftarrow h^{sk}$. $v \leftarrow h^n$. $c \leftarrow \mathcal{G}(m, pk, g, z, h, u, v)$. $s \leftarrow n + c \cdot sk \mod q$. $\sigma \leftarrow (z, c, s)$. Return σ. $Ver(pk, m, \sigma)$: $(z, c, s) \leftarrow \sigma$. $u' \leftarrow g^s \cdot pk^{-c}$. $h' \leftarrow \mathcal{H}(u')$. $v' \leftarrow h'^s \cdot z^{-c}$. $c' \leftarrow \mathcal{G}(m, pk, g, z, h', u', v')$. If $c = c'$, return $\top$. Else, return $\perp$.

Fig. 14: The Chevallier-Mames signature scheme.

quite similar to the key encapsulations output by CM-SCTK, with the main difference being that the randomizer u is computed as g^n rather than as ${sk_R}^n$ as in the signcryption tag-KEM. In Figure 15, the forger $\mathcal{F}$ uses a CM signature oracle $\mathcal{O}_{Sign}$ and random oracles $\mathcal{O}_G$ and $\mathcal{O}_H$ to simulate the runtime environment of a hypothetical adversary $\mathcal{A}$ against the signcryption tag-KEM. Three lists, L_G, L_H and L_{KDF}, are used to maintain state information.

$\mathcal{F}(pk; \mathcal{O}_{Sign}, \mathcal{O}_G, \mathcal{O}_H)$:
Form I and pk_S from pk.
$(sk_R, pk_R) \xleftarrow{R} Key_R(I)$.
$(E, \tau) \xleftarrow{R} \mathcal{A}(I, pk_S, sk_R, pk_R; \mathcal{O}_S, \mathcal{O}_E, \mathcal{G}_{sim}, \mathcal{H}_{sim}, KDF_{sim})$.
Return the signature $E = (c, r, s)$ on the message $\tau || pk_R$.

$\mathcal{O}_S$:
$K \xleftarrow{R} \mathcal{K}$.
Store K, overwriting any previous value.
Return K.

$\mathcal{O}_E(\tau)$:
If no stored K exists, return $\perp$.
Else read K and erase it from storage.
$(z, c, s) \xleftarrow{R} \mathcal{O}_{Sign}(\tau || pk_R)$.
$u \leftarrow (g^s \cdot {pk_S}^{-c})^{sk_R}$.
$h \leftarrow \mathcal{H}_{sim}(u)$.
$v \leftarrow h^s \cdot z^{-c}$.
Add $\big((\tau || pk_R, pk_S, g, z, h, u, v), c\big)$ to L_G.
Add (u, h) to L_H.
Add (u, K) to L_{KDF}.
Return (z, s, c).

$\mathcal{G}_{sim}(\tau || pk_R, pk_S, g, z, h, u, v)$:
Check if $\big((\tau || pk_R, pk_S, g, z, h, u, v), c\big)$ is in L_G.
If it is, return c.
Else, $u_{CM} \leftarrow u^{1/sk_R}$.
$c \leftarrow \mathcal{O}_G(\tau || pk_R, pk_S, g, z, h, u_{CM}, v)$.
Add $\big((\tau || pk_R, pk_S, g, z, h, u, v), c\big)$ to L_G.
Return c.

$\mathcal{H}_{sim}(u)$:
Check if (u, h) is in L_H.
If it is, return h.
Else, $u_{CM} \leftarrow u^{1/sk_R}$.
$h \leftarrow \mathcal{O}_H(u_{CM})$.
Add (u, h) to L_H.
Return h.

$KDF_{sim}(u)$:
Check if (u, K) is in L_{KDF}.
If it is, return K.
Else, $K \xleftarrow{R} \mathcal{K}$.
Add (u, K) to L_{KDF}.
Return K.

Fig. 15: Forgery algorithm $\mathcal{F}$.

To win the sUF-CMA game against CM-SCTK, $\mathcal{A}$ must return (τ, E) such that E is a valid CM signature on $\tau||pk_R$. Furthermore, $\mathcal{A}$ must not have received E as a response from the encapsulation oracle for the tag τ. If this is the case, then the forger $\mathcal{F}$ never queried the signing oracle on $\tau||pk_R$ and got E in return. Hence, $\mathcal{F}$ returns a valid sUF-CMA forgery of the CM signature scheme when $\mathcal{A}$ returns a valid sUF-CMA forgery of the CM signcryption tag-KEM.

As in the corresponding proof for Zheng-SCTK, the initial input given to $\mathcal{A}$, namely I, pk_S, sk_R, pk_R, are of the correct form and distributions. Furthermore, each of the oracles $\mathcal{O}_S$, $\mathcal{O}_E$, $\mathcal{G}_{sim}$, $\mathcal{H}_{sim}$ and KDF_{sim} output values that are internally consistent and from the correct spaces.

The only way that a simulation error may occur is if $\mathcal{O}_E$ adds an entry to one of the random oracle state lists that is inconsistent with previous entries. Since $\mathcal{O}_E$ explicitly uses $\mathcal{H}_{sim}$ to evaluate hashes, this is not a problem with respect to L_H. Furthermore, since $\mathcal{O}_{Sign}$ returns a valid signature on $\tau||pk_R$, any entry to L_G made by $\mathcal{O}_E$ will be consistent with previous entries made by $\mathcal{G}_{sim}$. However, since K is fixed by $\mathcal{O}_S$ before the value of u is determined, we are not guaranteed that L_{KDF} will remain consistent.

Consider an adversary that first asks q_{KDF} queries to the key derivation oracle, and then asks q_E queries to the symmetric key generation and encapsulation oracles. The value of u is equal to $g^{k \cdot sk_R}$ for some random $k \in \mathbb{Z}_q$, which means that u is uniformly distributed on G. Hence, the probability that the i'th encapsulation query causes an inconsistency in the KDF oracle is at most $\frac{q_{KDF}+i-1}{q}$. The probability of a simulation failure after q_E queries is thus at most

$$\frac{q_E(2q_{KDF} + q_E - 1)}{2q}, \tag{7}$$

and we may conclude that the advantage of $\mathcal{F}$ at creating sUF-CMA forgeries of CM signatures is negligibly close to that of $\mathcal{A}$ at forging the CM signcryption tag-KEM. □

One small result still needs to be established. This is because the original article only proves that the Chevallier-Mames signature scheme is UF-CMA secure [8]. Hence, a further argument is required to ensure that it is, in fact, also sUF-CMA secure.

Theorem 6. *The Chevallier-Mames signature scheme is sUF-CMA secure.*

Proof. The proof of this result can easily be adapted from the original proof of security [8]. Suppose a forger $\mathcal{F}$ outputs a message m and a signature (z, c, s), and let n, u, h and v be the internal values associated with that signature. The original proof of security for Chevallier-Mames signatures only used the fact that the forger outputs a message m that has not been queried to the signing oracle to ensure that the signing oracle never set the value of the output of the $\mathcal{G}$ oracle on the input (m, pk, g, z, h, u, v).

However, suppose that the forger $\mathcal{F}$ outputs a message m on which it *has* queried the signing oracle. Suppose further that this signing oracle query returns the signature (z', c', s') and that n', u', h' and v' are the internal values associated with this signature. If that query set the output of the $\mathcal{G}$ oracle on the input (m, pk, g, z, h, u, v), then it is clear that $z = z'$, $h = h'$, $u = u'$ and $v = v'$. We may also conclude that $c = c'$ as c' is defined to be the output of the $\mathcal{G}$ oracle. So now we know

$$g^s \cdot pk^{-c} = u = u' = g^{s'} \cdot pk^{-c} \tag{8}$$

and so $s = s' \bmod q$. Hence, $(z', c', s') = (z, c, s)$. Therefore, the only way that the signing oracle could have set the output of the $\mathcal{G}$ oracle on (m, pk, g, z, h, u, v) is if the singing oracle was queried on the input m and returned the signature (z, c, s). This means that if the forger $\mathcal{F}$ wins the sUF-CMA game, then the signing oracle could not have set the output of the $\mathcal{G}$ oracle on this input (m, pk, g, z, h, u, v). Once we have established this, the original proof of security of Chevallier-Mames proves that the scheme is, in fact, sUF-CMA secure. □

It should be noted that a similar proof for the sUF-CMA security of the Chevallier-Mames signature scheme was developed independently by Chevallier-Mames [9].

B.2 IND-CCA2 security of CM-SCTK

We will use standard techniques to show that the CM-SCTK is IND-CCA2. If we challenge the attacker to distinguish whether the key K^* is encapsulated by the challenge encapsulation (z^*, c^*, s^*), and we model the key derivation function *KDF* as a random oracle, then the only way that the attacker can have any advantage is by querying the *KDF* oracle on the input u^* associated with the signature. We arrange the input values so that this value is the solution to a CDH problem. However, in order to simulate all the oracles to which the attacker has access, we will need to have access to a DDH oracle. Hence, we reduce the IND-CCA2 security of the CM-SCTK to the GDH problem.

Suppose we wish to solve a given instance of the GDH problem, i.e. we are given $X = g^x$ and $Y = g^y$, and we wish to find g^{xy}. Since the value $u^* = {pk_R}^{n^*}$, we set $pk_R = Y$ and $g^{n^*} = X$. Now, from the specification of the verification algorithm, we know that

$$X^{sk_R} = u^* = (g^{s^*} \cdot {pk_S}^{-c^*})^{sk_R} \qquad \text{and so} \qquad pk_S = (X/g^{s^*})^{-1/c^*} .$$

Therefore, we choose s^* and c^* at random from $\mathbb{Z}_q$, and set pk_S as above. Furthermore, we randomly choose α^* at random from $\mathbb{Z}_q$ and set $h^* = \mathcal{H}(u^*) = g^{\alpha^*}$. We may now set $z^* = {pk_S}^{\alpha^*}$ and $v^* = X^{\alpha^*}$. This gives us a completely consistent challenge signcryption provided we make sure that we answer the oracle queries $\mathcal{H}(u^*)$ and $\mathcal{G}(\tau^* || pk_R, pk_S, g, z, h, u, v)$ correctly. Furthermore, all of the variables are chosen from precisely the correct distributions. The specification of the GDH solving algorithm is given in Figure 16.

We need to simulate the attacker access to the $\mathcal{G}$, $\mathcal{H}$, *KDF* oracles, as well as the *Sym*, *Encap* and *Decap* oracles. We simulate direct queries to the $\mathcal{G}$, $\mathcal{H}$ and *KDF* oracles in a simple way, by generating responses to new queries at random from the appropriate range and storing the outputs in a set of lists L_1^G, L_1^H and L_1^{KDF}. We use a second set of lists (L_2^G, L_2^H and L_2^{KDF}) to store the

GDH-solver$(I, X, Y; \mathcal{O}_{DDH})$:
$(G, q, g) \leftarrow I$.

$c^* \xleftarrow{R} \mathbb{Z}_q$.
$pk_S \leftarrow (X/g^{s^*})^{c^*}$.
$\alpha^* \xleftarrow{R} \mathbb{Z}_q$.
$z^* \leftarrow {pk_S}^{\alpha^*}$.
$K^* \xleftarrow{R} \mathcal{K}$.

$s^* \xleftarrow{R} \mathbb{Z}_q$.
$pk_R \leftarrow Y$.
$h^* \leftarrow g^{\alpha^*}$.
$v^* \leftarrow X^{\alpha^*}$.
$E^* \leftarrow (z^*, c^*, s^*)$.

$state_1 \xleftarrow{R} \mathcal{A}_1(I, pk_S, pk_R; \mathcal{O}_S, \mathcal{O}_E, \mathcal{O}_D, \mathcal{O}_G, \mathcal{O}_H, \mathcal{O}_{KDF})$.
$(state_2, \tau^*) \xleftarrow{R} \mathcal{A}_2(state_1, K^*; \mathcal{O}_S, \mathcal{O}_E, \mathcal{O}_D, \mathcal{O}_G, \mathcal{O}_H, \mathcal{O}_{KDF})$.
$b' \xleftarrow{R} \mathcal{A}_3(state_2, E^*; \mathcal{O}_S, \mathcal{O}_E, \mathcal{O}_D, \mathcal{O}_G, \mathcal{O}_H, \mathcal{O}_{KDF})$.

If the GDH solution has not been found, return a random group element.

Fig. 16: Specification of GDH-solving algorithm.

values that oracles must take in order to be consistent with the *Encap* oracle. The specifications of the $\mathcal{G}$, $\mathcal{H}$ and *KDF* oracles are given in Figure 17.

Next we turn our attention to the symmetric and encapsulation oracles. These are detailed in Figure 18. This simulation is perfectly consistent provided that the encapsulation algorithm doesn't add an entry to the L_2^G list which is inconsistent with a previous $\mathcal{G}$ oracle query. In any single encapsulation oracle query, the c and s values are chosen at random; hence, we know that ϕ is uniformly distributed over the group G. Therefore, the probability that the entry $(\tau || pk_R, g, z, h, \phi^{sk_R}, v)$ has been set in L_1^G by one of the direct $\mathcal{G}$ oracle query is at most q_G/q. If the entry $(\tau || pk_R, g, z, h, \phi, v)$ has been set in L_2^G by a previous encapsulation oracle query, then either the c we have randomly chosen is consistent, or we have found two values (c, s) and (c', s') for which the computed ϕ value is the same. If this has happened, then

$$g^s \cdot {pk_S}^{-c} = \phi = g^{s'} \cdot {pk_S}^{-c'}$$

and so the discrete logarithm of pk_S is $(s - s')/(c - c')$. From here we may recover x and so solve the GDH problem. Therefore, the probability that the encapsulation oracle is inconsistent with previous entries is (after q_E encapsulation oracle queries) bounded above by $q_E q_G/q$.

Lastly, we turn to the decapsulation algorithm. This algorithm is specified in Figure 19. This algorithm perfectly simulates the decryption algorithm unless we reject some valid signcryption of the form (z, c^*, s^*) and tag τ. We break this into two cases: the case where such a query is first made before the challenge is issued and the case where such a query is first made after the challenge is issued. Before the challenge is issued, the attacker has no information about c^* and s^*, and so the probability that the attacker queries the decapsulation oracle on this input is bounded above by q_D/q^2. The situation becomes more complex after

$\mathcal{O}_{KDF}(u)$:
If $\top \leftarrow \mathcal{O}_{DDH}(X, Y, u)$.
Found the GDH solution!
Else if (u, K) is in L_1^{KDF}:
Return K.
Else if $\top \leftarrow \mathcal{O}_{DDH}(Y, \phi, u)$ for some (ϕ, K) in L_2^{KDF}:
Return K.
Else $K \xleftarrow{R} \mathcal{K}$.
Append (u, K) to L_1^{KDF}.
Return K.

$\mathcal{O}_H(u)$:
If $\top \leftarrow \mathcal{O}_{DDH}(X, Y, u)$.
Found the GDH solution!
Else if (u, α, h) is in L_1^H:
Return h.
Else if $\top \leftarrow \mathcal{O}_{DDH}(Y, \phi, u)$ for some (ϕ, α, h) in L_2^H:
Return h.
Else $\alpha \xleftarrow{R} \mathbb{Z}_q$.
$h \leftarrow g^\alpha$.
Append (u, α, h) to L_1^H.
Return h.

$\mathcal{O}_G(\tau || pk_R, pk_S, g, z, h, u, v)$:
If $\top \leftarrow \mathcal{O}_{DDH}(X, Y, u)$.
Found the GDH solution!
Else if $\big((\tau || pk_R, pk_S, g, z, h, u, v), c\big)$ is in L_1^G:
Return c.
Else if $\big((\tau || pk_R, pk_S, g, z, h, \phi, v), c\big)$ is in L_2^G and $\top \leftarrow \mathcal{O}_{DDH}(Y, \phi, u)$:
Return c.
Else $c \xleftarrow{R} \mathbb{Z}_q$.
Append $\big((\tau || pk_R, pk_S, g, z, h, u, v), c\big)$ to L_1^G.
Return c.

Fig. 17: Specification of random oracle simulators.

$\mathcal{O}_S$:
$c \xleftarrow{R} \mathbb{Z}_q$.
$s \xleftarrow{R} \mathbb{Z}_q$.
$\phi \leftarrow g^s \cdot {pk_S}^{-c}$.
If $\top \leftarrow \mathcal{O}_{DDH}(\phi, Y, u)$ for some (u, K') in L_1^{KDF} or if (ϕ, K') is in L_2^{KDF} then $K \leftarrow K'$.
Else $K \xleftarrow{R} \mathcal{K}$.
Append (ϕ, K) to L_2^{KDF}.
$\omega \leftarrow (c, s, \phi, K)$.
Store ω, overwriting any previous value.
Return K.

$\mathcal{O}_E(\tau)$:
If no stored ω exists, return $\perp$.
Else read ω from storage and erase it.
$(c, s, \phi, K) \leftarrow \omega$.
If $\top \leftarrow \mathcal{O}_{DDH}(\phi, Y, u)$ for some (u, α, h') is in L_1^H or if (ϕ, α, h') in L_2^H then $h \leftarrow h'$.
Else $\alpha \xleftarrow{R} \mathbb{Z}_q$, $h \leftarrow g^\alpha$ and append (ϕ, α, h) to L_2^H.
$z \leftarrow {pk_S}^\alpha$.
$v \leftarrow \phi^\alpha$.
$E \leftarrow (z, c, s)$.
If there exists $(c', s', \phi, z', \tau', K')$ in L_{Encap} such that $(c, s) \neq (c', s')$ then we have found the GDH solution!
Else append $\big((\tau || pk_R, pk_S, g, z, h, \phi, v), c\big)$ to L_2^G.
Append (c, s, ϕ, z, τ, K) to L_{Encap}.
Return E.

Fig. 18: Specification of symmetric key generation and encapsulation oracles.

the challenge has been issued. In this case, we know that $z \neq z^*$ but

$$\mathcal{G}(\tau || pk_R, pk_S, g, z, h, u, v) = c^* \,.$$

This means that either there is an entry of the form $((\tau||pk_R, pk_S, g, z, h, u, v), c^*)$ in L_1^G, or an entry of the form $((\tau||pk_R, pk_S, g, z, h, u^{\frac{1}{sk_R}}, v), c^*)$ in L_2^G or that $\mathcal{G}(\tau||pk_R, pk_S, g, z, h, u, v) = c^*$ even though the attacker has not queried the $\mathcal{G}$ oracle on this input (either implicitly or explicitly) yet. For a single decapsulation oracle query, the probability that this occurs is bounded by $(q_G + q_E + q_D + 1)/q$. Hence, the probability that this occurs at all in the simulation is bounded by $q_D(q_G + q_E + q_D + 1)/q$.

$\mathcal{O}_D(E, \tau)$:
$(z, c, s) \leftarrow E$.
If $(c, s) = (c^*, s^*)$ then return $\perp$.
If there exists $(c, s, \phi', z, \tau, K)$ on L_{Encap} for some ϕ' and K, then return K.
$\phi \leftarrow g^s \cdot {pk_S}^{-c}$.
If $\top \leftarrow \mathcal{O}_{DDH}(\phi, Y, u)$ for some (u, α, h) in L_1^H or if (ϕ, α, h) in L_2^H for some α and h, then recover h and α from the appropriate list.
Else $\alpha \xleftarrow{R} \mathbb{Z}_q$, $h \leftarrow g^\alpha$ and append (ϕ, α, h) to L_2^H.
$v \leftarrow h^s \cdot z^{-c}$.
If $\top \leftarrow \mathcal{O}_{DDH}(\phi, Y, u)$ for some $((\tau||pk_R, pk_S, g, z, h, u, v), c')$ in L_1^G or if $((\tau||pk_R, pk_S, g, z, h, \phi, v), c')$ in L_2^G for some c', then recover c' from the appropriate list.
Else $c' \xleftarrow{R} \mathbb{Z}_q$ and append $((\tau||pk_R, pk_S, g, z, h, \phi, v), c')$ to L_2^G.
If $c' \neq c$ then return $\perp$.
If $\top \leftarrow \mathcal{O}_{DDH}(\phi, Y, u)$ for some (u, K) in L_1^{KDF} or if (ϕ, K) in L_2^{KDF} for some K, then recover K.
Else $K \xleftarrow{R} \mathcal{K}$ and append (ϕ, K) to L_2^{KDF}.
Return K.

Fig. 19: Specification of the decapsulation oracle.

If we collect all the error terms we obtain the bound

$$\epsilon_{IND} \leq \epsilon_{GDH} - \frac{q_G q_E}{q} - \frac{q_D}{q^2} - \frac{q_D(q_G + q_E + q_D + 1)}{q} \tag{9}$$

where ϵ_{IND} and ϵ_{GDH} are the attacker's advantage in breaking the IND-CCA2 security of the CM-SCTK and the probability that the algorithm we described solves the GDH problem respectively. □

C Proof of Theorem 5

The intuition behind Theorem 5 is quite simple. By construction, the signcryption tag-KEM does as little as possible. An adversary should have no room to do anything interesting with the scheme, without having to break the underlying signcryption scheme (which is assumed to be secure). To confirm this intuition, we construct generic sUF-CMA and IND-CCA2 adversaries against signcryption schemes with associated data, that use adversaries against the derived SCTK as subroutines.

Signcryption schemes with associated data behave as one would expect; the only difference in syntax is that an additional parameter representing the plaintext data, denoted τ, is given as additional input to the signcryption and unsigncryption algorithms. An adversary against confidentiality should produce two messages of the same length as well as a plaintext tag, and distinguish which message has been signcrypted under that tag. An adversary against authenticity and integrity should produce a message, an signcryptext and a tag, such that the signcryptext and tag unsigncrypt to that message.

C.1 sUF-CMA security of the construction

With respect to the authenticity and integrity of the construction, we show that an efficient sUF-CMA adversary $\mathcal{A}$ against the signcryption tag-KEM can be used to construct an efficient sUF-CMA adversary $\mathcal{B}$ against the underlying signcryption scheme.

A successful adversary against the SCTK returns an encapsulation E and a tag τ, so that the unsigncryption algorithm returns a key K rather than the error symbol $\perp$. It is straightforward to verify that the resulting key, encapsulation and tag will in fact be a valid forgery of the underlying signcryption scheme $\mathcal{SC}$. Figure 20 gives a complete specification of $\mathcal{B}$.

$\mathcal{B}(I, pk_S, sk_R, pk_R; \mathcal{O}_{SC})$:
$(E, \tau) \stackrel{R}{\leftarrow} \mathcal{A}(I, pk_S, sk_R, pk_R; \mathcal{O}_S, \mathcal{O}_E)$.
$K \leftarrow USC(pk_S, sk_R, E, \tau)$.
Return (K, E, τ).

$\mathcal{O}_S$:	$\mathcal{O}_E(\tau)$:
$K \stackrel{R}{\leftarrow} \mathcal{K}$.	If no stored K exists, return $\perp$.
Store K, overwriting any previous value.	Else read K and erase it from storage.
Return K.	$E \stackrel{R}{\leftarrow} \mathcal{O}_{SC}(K, \tau)$. Return E.

Fig. 20: sUF-CMA adversary against $\mathcal{SC}$.

In the simulation in Figure 20, $\mathcal{B}$ wins if $\mathcal{O}_{SC}$ never answered E on the query (K, τ). Meanwhile, $\mathcal{A}$ wins the sUF-CMA game against SCTK whenever $\mathcal{O}_E$ has not answered E on the question τ. From the specification, it follows that $\mathcal{O}_{SC}$ has not answered E on the query $(\cdot, \tau)$ for *any* key K. Hence $\mathcal{B}$ wins whenever $\mathcal{A}$ does, which implies that the signcryption tag-KEM is sUF-CMA secure relative to $\mathcal{SC}$. $\square$

C.2 IND-CCA2 security of the construction

It is similarly straightforward to construct a convincing security proof for the IND-CCA2 security of signcryption tag-KEMs derived from signcryption schemes with supported associated data. As specified, the symmetric key generation algorithm simply picks a random key, whereupon the encapsulation algorithm signcrypts it. The supplied tag τ is only used as associated data for the signcryption algorithm. This means that an efficient IND-CCA2 adversary against SCTK must distinguish whether the challenge encapsulation corresponds to the signcryption of the supplied key K_b, together with the associated data τ.

Figure 21 specifies an adversarial algorithm $\mathcal{B}$ that uses such an IND-CCA2 adversary $\mathcal{A}$ against the signcryption tag-KEM to break the IND-CCA2 security of the underlying signcryption scheme $\mathcal{SC}$[5].

$\mathcal{B}_1(I, pk_S, pk_R; \mathcal{O}_{SC}, \mathcal{O}_{USC})$: $state_1 \xleftarrow{R} \mathcal{A}_1(I, pk_S, pk_R; \mathcal{O}_S, \mathcal{O}_E, \mathcal{O}_D)$. $K_0 \xleftarrow{R} \mathcal{K}$. $K_1 \xleftarrow{R} \mathcal{K}$. $(state_2, \tau^*) \xleftarrow{R} \mathcal{A}_2(state_1, K_0; \mathcal{O}_S, \mathcal{O}_E, \mathcal{O}_D)$. Return $(K_0, K_1, \tau^*, state_2)$. $\mathcal{B}_2(state_2, \sigma^*; \mathcal{O}_{SC}, \mathcal{O}_{USC})$: $b' \xleftarrow{R} \mathcal{A}_3(state_2, \sigma^*)$. Return b'.	$\mathcal{O}_S$: $K \xleftarrow{R} \mathcal{K}$. Store K, overwriting any previous value. Return K. $\mathcal{O}_E(\tau)$: If there exists no stored K: Return $\perp$ and terminate. Else: read K and erase it. $E \xleftarrow{R} \mathcal{O}_{SC}(K, \tau)$. Return E. $\mathcal{O}_D(E, \tau)$: Return $\mathcal{O}_{USC}(\sigma, \tau)$.

Fig. 21: IND-CCA2 adversary against $\mathcal{SC}$

Let b be the hidden bit that $\mathcal{B}$ is attempting to guess in the IND-CCA2 game against $\mathcal{SC}$. By the construction in Figure 21, it follows that $\mathcal{A}$ receives a valid encapsulation if $b = 0$, and a random encapsulation otherwise. Hence, if $\mathcal{A}$ has any advantage in the IND-CCA2 game against SCTK, then $\mathcal{B}$ will have the same advantage attacking the original signcryption scheme $\mathcal{SC}$.

From this and the previous result we conclude that the signcryption tag-KEM SCTK is secure whenever the signcryption scheme $\mathcal{SC}$ on which it is built is secure. $\square$

[5] On a technical note, it is necessary to assume that $\mathcal{B}$ chooses a representation of the keyspace $\mathcal{K}$ for which all $K \in \mathcal{K}$ are of equal length. This is unproblematic.