Hybrid Cryptography

Alexander W. Dent

November 15, 2004

a.dent@rhul.ac.uk

Information Security Group,
Royal Holloway, University of London
Egham Hill, Egham, Surrey, TW20 0EX,
UNITED KINGDOM.

Contents

1 Introduction

The ISO/IEC JTC1/SC27 standardisation committee [23] suggest that hybrid cryptography can be defined as the branch of asymmetric cryptography that makes use of convenient symmetric techniques to remove some of the problems inherent in normal asymmetric cryptosystems (e.g., the problems encounter when trying to process long messages quickly). This definition is somewhat vague. What makes certain mathematical manipulations "asymmetric" and others "symmetric"? Why are S-boxes considered a symmetric technique and elliptic curves an asymmetric technique? And what should finite field arithmetic, which is used in both asymmetric and symmetric algorithms, be classed as?

It is perhaps better to define hybrid cryptography as the branch of asymmetric cryptography that makes use of keyed symmetric cryptosystems as black-box algorithms with certain security properties. The critical point of this definition is that it is the properties of the symmetric cryptosystem that are used to construct the asymmetric scheme, rather than the technical details about the way in which the symmetric algorithm achieves these security properties. We specify the use of keyed symmetric algorithms to make sure that an asymmetric cryptosystems that makes use of hash functions (as almost all asymmetric cryptosystems seem to do) are not automatically classed as hybrid schemes.

Traditionally, hybrid cryptography has been concerned with building asymmetric encryption schemes (see Section 2). In these cryptosystems a symmetric encryption scheme is used to overcome the problems typically associated with encrypting long messages using "pure" asymmetric techniques. More recently, symmetric encryption algorithms have been used to solve the same problem in signcryption schemes [2, 18] (see Section 3).

Another recent advance in hybrid cryptography is the development of the KEM–DEM model for hybrid encryption algorithms [15, 31]. This model splits a hybrid encryption scheme into two distinct components: an asymmetric key encapsulation mechanism (KEM) and a symmetric data encapsulation mechanisms (DEM). Whilst the KEM–DEM model does not model all possible hybrid encryption schemes, and there are several examples of hybrid encryption schemes that do not fit into the KEM–DEM model, it does have the advantage of allowing the security requirements of the asymmetric and symmetric parts of the scheme to be completely separated and studied independently.

Shoup's KEM–DEM model demonstrates what should be an overriding principle of hybrid cryptography: it is not necessary for an asymmetric scheme to fully involve itself in the details of providing a security service — the security service can be provided by a symmetric scheme provided the asymmetric scheme is in full control of that process (say, by generating the secret key that the symmetric scheme uses). Hence, we can fully separate

the asymmetric and symmetric parts of the scheme.

This paper examines the ways that this "separation principle" can be used to construct different types of hybrid asymmetric schemes by making use of different types of symmetric cryptosystem. We will begin by reviewing hybrid encryption in Section 2. Next we examine the recent trends in signcryption, noting that there are two types of signcryption schemes, and apply the separation principle to signcryption schemes with outsider security (Section 3). We then show that it is impossible to have an efficient hybrid signature scheme (Section 4) but that, despite their similarities, it is possible to build an efficient hybrid signcryption scheme with insider security (Section 5). Lastly, as we have only considered the security of cryptographic schemes in the two-user setting, we examine the problems associated with extending the proposed schemes into a multi-user setting (Section 6).

2 Hybrid Encryption Schemes

An encryption scheme is meant to provide a confidentiality service. Hence it is natural to try and build a hybrid encryption scheme using a symmetric encryption scheme under the control of some asymmetric process. In this section we will review the notion of an asymmetric encryption scheme, the KEM–DEM construction and the security results associated with this construction.

2.1 Asymmetric encryption schemes

Formally, an asymmetric encryption scheme is a triple of algorithms:

1. A probabilistic *key generation algorithm*, $\mathcal{G}$, which takes as input a security parameter[1] 1^k and outputs a public/private key pair (pk, sk). The public key defines the *message space* $\mathcal{M}$, which is the set of all possible messages which can be submitted to the encryption algorithm, and the *ciphertext space* $\mathcal{C}$, which is set of possible ciphertexts that can be submitted to the decryption algorithm.

2. A (possibly) probabilistic *encryption algorithm*, $\mathcal{E}$, which takes as input a message $m \in \mathcal{M}$ and a public key pk and outputs a ciphertext $C \in \mathcal{C}$. We will denote this as $C = \mathcal{E}(pk, m)$.

[1]The term 'security parameter' is often used to refer to both k and 1^k. Generally speaking, we wish algorithms to run in time at most bounded by a polynomial in k but rather than specify this directly, we instead specify that our algorithms should run in time bounded above by a polynomial in their input size and provide 1^k as an input. This allows us to cope with algorithms which can take arbitrarily long messages and yet should still run in time bounded by a polynomial in k for all reasonable (polynomial-length) messages.

3. A deterministic *decryption algorithm*, $\mathcal{D}$, which takes as input a ciphertext $C \in \mathcal{C}$ and a secret key sk and outputs either a message $m \in \mathcal{M}$ or the error symbol $\perp$. We denote this as $m = \mathcal{D}(sk, C)$.

It is important that any asymmetric encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ has the correctness property, i.e. that for almost all valid key pairs (pk, sk), the decryption $D(sk, C) = m$ for almost all encryptions $C = \mathcal{E}(pk, m)$.

It is also important that an asymmetric encryption scheme satisfy some kind of security property. A full discussion of the different types of security property that an asymmetric encryption scheme may aspire to is given by Bellare, Desai, Pointcheval and Rogaway [8]. For our purposes it is sufficient to define the notion of IND security. This notion of security suggests that a scheme is secure if an attacker's advantage in breaking the scheme is *negligible* as a function of the security parameter k.

Definition 1 (Negligible function) *A function $f : \mathbb{Z} \to \mathbb{R}$ is negligible if for every polynomial p there exists an integer N_p such that $|f(n)| \leq 1/p(n)$ for all $n \geq N_p$.*

Almost all of the security models we shall define our phrased in terms of a game played between a hypothetical challenger and an attacker. The game runs in two stages: a pre-challenge "find" stage and a post-challenge "guess" stage. Hence, an attacker is best considered to be a pair of probabilistic Turing machines $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$.

For a given security parameter k, the IND-CCA2 game runs as follows:

1. The challenger generates a valid key-pair (pk, sk) by running the key generation algorithm $\mathcal{G}(1^k)$.

2. The attacker runs $\mathcal{A}_1$ on the input pk. The algorithm $\mathcal{A}_1$ terminates by outputting a pair of (equal length) messages (m_0, m_1), as well as some state information *state*. During any point in its execution, $\mathcal{A}_1$ may query a decryption oracle that will, when given a ciphertext C, return $\mathcal{D}(sk, C)$.

3. The challenger picks a bit $b \in \{0, 1\}$ uniformly at random, and forms the challenge ciphertext $C^* = \mathcal{E}(pk, m_b)$.

4. The attacker runs $\mathcal{A}_2$ on the input $(C^*, state)$. This algorithm outputs a guess b' for b. Again, at any point during its execution, $\mathcal{A}_2$ may query a decryption oracle that will, given a ciphertext $C \neq C^*$, return $\mathcal{D}(sk, C)$.

The attacker wins the game if $b' = b$. The attacker's advantage is defined to be:

$$|Pr[b = b'] - 1/2| \,. \tag{1}$$

Definition 2 (IND Security) *An asymmetric encryption scheme is secure in the IND-CCA2 attack model (or IND-CCA2 secure) if, for all polynomial-time attackers $\mathcal{A}$, the advantage of an $\mathcal{A}$ in winning the IND-CCA2 game is negligible as a function of the security parameter k.*

2.2 Symmetric schemes

In this section we will briefly review symmetric encryption schemes, MAC schemes, and their associated security notions.

For our purposes, a symmetric encryption scheme is a pair of algorithms $(\textsc{Enc}, \textsc{Dec})$. The encryption algorithm, $\textsc{Enc}$, takes as input a message $m \in \{0,1\}^*$ of any length and a key K of length $EncKeyLen$; and outputs a ciphertext $C = \textsc{Enc}_K(m)$. The decryption algorithm, $\textsc{Dec}$, takes as input a ciphertext $C \in \{0,1\}^*$ and a key K of length $EncKeyLen$; and outputs either a message $m = \textsc{Dec}_K(C)$ or the error symbol $\perp$. We will assume that the key length $EncKeyLen$ depends upon a security parameter k.

We require that any symmetric encryption scheme satisfies the soundness property: for any key K of length $EncKeyLen$ and message $m \in \{0,1\}^*$ we have that $\textsc{Dec}_K(\textsc{Enc}_K(m)) = m$.

We also require that a symmetric encryption scheme satisfy some notion of security. We give two notions of security, both of which are very similar to the IND-CCA2 notion of security for asymmetric encryption schemes given in Section 2.1. These are known as IND-PA and IND-CCA2 security[2][3].

For IND-PA security, the attack is again phrased in terms of a game between a challenger and a two-stage attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. The IND-PA game runs as follows:

1. The challenger randomly generates a symmetric key K of length $EncKeyLen$.
2. The attacker runs $\mathcal{A}_1$ on the input 1^k. The algorithm $\mathcal{A}_1$ terminates by outputting a pair of (equal length) messages (m_0, m_1), as well as some state information *state*.
3. The challenger chooses a bit $b \in \{0,1\}$ uniformly at random, and forms the challenge ciphertext $C^* = \textsc{Enc}_K(m_b)$.
4. The attacker runs $\mathcal{A}_2$ on the input $(C^*, state)$. This algorithm outputs a guess b' for b.

The attacker wins the game if $b = b'$.

[2]Here "PA" stands for "passive attack", signifying that the attacker has no access to any kind of encryption or decryption oracle, and "CCA" stands for chosen ciphertext attack, signifying that the attacker has access to a decryption oracle.

[3]We abuse the standard terminology a bit here: the definition of IND-CCA2 given here is what Cramer and Shoup [15] calls one-time symmetric key encryption. It is different from IND-CCA2 security as defined by Bellare et al [7].

For IND-CCA2 security, the attack is very similar. This time the game is as follows.

1. The challenger randomly generates a symmetric key K of length $EncKeyLen$.
2. The attacker runs $\mathcal{A}_1$ on the input 1^k. The algorithm $\mathcal{A}_1$ terminates by outputting a pair of (equal length) messages (m_0, m_1), as well as some state information *state*.
3. The challenger chooses a bit $b \in \{0,1\}$ uniformly at random, and forms the challenge ciphertext $C^* = \textsc{Enc}_K(m_b)$.
4. The attacker runs $\mathcal{A}_2$ on the input $(C^*, state)$. This algorithm outputs a guess b' for b. During its execution, $\mathcal{A}_2$ is allowed access to a decryption oracle that will, when given a ciphertext $C \neq C^*$, return $\textsc{Dec}_K(C)$.

The attacker wins the game if $b = b'$.

In both cases, the attacker's advantage of winning the game is defined to be:

$$|Pr[b = b'] - 1/2|\,. \tag{2}$$

Definition 3 (IND security for a symmetric encryption scheme) *A symmetric encryption scheme is said to be IND-PA secure if, for all polynomial-time attackers $\mathcal{A}$, the advantage that $\mathcal{A}$ has in winning the IND-PA game is negligible as a function of the security parameter k.*

Similarly, a symmetric scheme is said to be IND-CCA2 secure if, for all polynomial-time attackers $\mathcal{A}$, the advantage that $\mathcal{A}$ has in winning the IND-CCA2 game is negligible as a function of the security parameter k.

Next we move onto MAC schemes. A MAC scheme is simply a deterministic algorithm MAC that takes as input a message $m \in \{0,1\}^*$ of any length and a key K of length $MACKeyLen$; and outputs a MAC tag τ of length $MACTagLen$. We assume that both the MAC key length $MACKeyLen$ and the MAC tag length $MACTagLen$ depend upon some security parameter k.

Again we define the security of the MAC algorithm in terms of a game played between a hypothetical challenger and an attacker. For a MAC algorithm the game runs as follows:

1. The challenger random generates a key K of length $MACKeyLen$.
2. The attacker submits a message m_0 to the challenger.
3. The challenger computes the MAC of m_0, $\tau = \text{MAC}_K(m_0)$, and passes this back to the attacker.
4. The attacker then outputs any number of pairs (m_i, τ_i) where $m_i \neq m_0$.

The attacker is said to win the game if any output pair (m_i, τ_i) satisfies $\tau_i = \mathrm{MAC}_K(m_i)$.

Definition 4 (MAC security) *A MAC scheme is said to be secure if, for all polynomial-time attackers $\mathcal{A}$, the probability that $\mathcal{A}$ can win the above security game is negligible as a function of the security parameter k.*

2.3 KEM–DEM hybrid encryption schemes

If we define a hybrid encryption scheme as an asymmetric encryption scheme which makes use of some keyed symmetric cryptography as a "black-box" then it is very difficult to make any statements about the security of hybrid encryption schemes as there are far too many ways in which the symmetric scheme could be used within the cryptosystem. However, almost all the practical examples of hybrid encryption schemes [1, 15, 19, 25, 27] are based on a simple idea:

1. an asymmetric section generates a suitable symmetric key and encrypts that key using an asymmetric encryption scheme, and

2. a symmetric section encrypts the message using the randomly generated symmetric key.

Shoup [31] has shown that these two parts of a hybrid encryption scheme can be separated and security criteria defined for both sections individually. The asymmetric section is known as the *key encapsulation mechanism* or KEM. The symmetric section is known as the *data encapsulation mechanism* or DEM. We will refer to any hybrid encryption scheme constructed from these ideas as a KEM–DEM construction. It should be noted that not every hybrid encryption scheme can necessarily be phrased as a KEM–DEM construction: whilst ECIES [1] can be viewed as a KEM–DEM construction, EPOC-2 [27] can not.

A KEM is a triple of algorithms consisting of:

1. The key generation algorithm, *Gen*, which takes as input a security parameter 1^k and outputs a public/private key pair (pk, sk).

2. A probabilistic encapsulation algorithm, *Encap*, which takes as input a public key pk and outputs a key K and an encapsulation of that key C. We denote this as $(K, C) = \mathit{Encap}(pk)$.

3. A deterministic decapsulation algorithm, *Decap*, which takes as inputs the private key pk and an encapsulation C, and outputs a symmetric key K or the error symbol $\perp$. We denote this as $K = \mathit{Decap}(sk, C)$.

Hence, KEMs are very similar to asymmetric encryption schemes; the only difference is that, unlike an encryption algorithm, the KEM's encapsulation

algorithm does not take any form of message as input but rather randomly generates its own "message" — the symmetric key K. Just as with an asymmetric encryption scheme, it is important that a KEM satisfies a correctness property, i.e. that for almost all public/private key pair (pk, sk) we have that $K = \mathit{Decap}(C, sk)$ for almost all $(K, C) = \mathit{Encap}(pk)$.

A DEM is a pair of algorithms consisting of:

1. A deterministic encryption algorithm, Enc, which takes as input a message $m \in \{0,1\}^*$ of any length and a symmetric key K of some pre-determined length. It outputs an encryption $C = \textsc{Enc}_K(m)$.

2. A deterministic decryption algorithm, Dec, which takes as input an encryption $C \in \{0,1\}^*$ and a symmetric key K of some pre-determined length, and outputs either a message $m \in \{0,1\}^*$ or the error symbol $\perp$.

Hence, DEMs are very similar to symmetric encryption schemes. Again, a DEM must satisfy a correctness property: that for every key K of the correct length, $m = \textsc{Dec}_K(\textsc{Enc}_K(m))$.

It should now be easy to see that a KEM and a DEM can be 'slotted together' to form an asymmetric encryption algorithm. If $(\mathit{Gen}, \mathit{Encap}, \mathit{Decap})$ is a KEM and $(\textsc{Enc}, \textsc{Dec})$ is a DEM, and for any security parameter k the length of the symmetric keys that are output by the KEM is equal to the length of the symmetric keys taken as input by the DEM, then we can form an asymmetric encryption scheme as follows:

- The key generation algorithm is given by Gen.

- The encryption algorithm for a message m under a public key pk is given by:

  1. Set $(K, C_1) = \mathit{Encap}(pk)$.
  2. Set $C_2 = \textsc{Enc}_K(m)$.
  3. Output (C_1, C_2).

- The decryption algorithm for a ciphertext $C = (C_1, C_2)$ under a private key sk is given by:

  1. Set $K = \mathit{Decap}(sk, C_1)$. If $K = \perp$ then output $\perp$ and stop.
  2. Set $m = \textsc{Dec}_K(C_2)$. If $m = \perp$ then output $\perp$ and stop.
  3. Output m.

2.4 The security criterion for a KEM

The beauty of the KEM–DEM construction is that, as we shall see, it allows us to consider KEMs and DEMs separately. Each of the two parts of the construction has its own security criteria, independent of the operation of the other. The security criteria for a KEM is called "output indistinguishability" or, sometimes, "real-or-random indistinguishability".

The security of a KEM is phrased in terms of a game played between a hypothetical challenger and two-stage attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. The attack goal for a KEM is to distinguish the real key corresponding to an encapsulation from a randomly generated key. This is known as the IND-CCA2 game and, for a given security parameter k, works as follows:

1. The challenger generates a valid public/private key pair (pk, sk) by running $Gen(1^k)$.

2. The attacker runs $\mathcal{A}_1$ on the input pk. It terminates by outputting some state information *state*. During its execution $\mathcal{A}_1$ may query a decapsulation oracle that will, when given an encapsulation C, return $Decap(sk, C)$.

3. The challenger generates a valid encapsulation (K_0, C^*) by running $Encap(pk)$. It also generates a random key K_1 of the same length as K_0. Next it chooses a bit $b \in \{0, 1\}$ uniformly at random and sets $K^* = K_b$. The challenge encapsulation is (K^*, C^*).

4. The attacker runs $\mathcal{A}_2$ on the input (K^*, C^*) and *state*. It terminates by outputting a guess b' for b. Again, during its execution $\mathcal{A}_2$ may query a decapsulation oracle that will, when given an encapsulation $C \neq C^*$, return $Decap(sk, C)$.

The attacker wins the game if $b = b'$. The attacker's advantage is defined to be:

$$|Pr[b = b'] - 1/2| \,. \tag{3}$$

Definition 5 (Security of a KEM) *A KEM is secure in the IND-CCA2 security model if, for all polynomial-time attackers $\mathcal{A}$, the advantage that $\mathcal{A}$ has in breaking the IND-CCA2 game is negligible as a function of the security parameter k.*

Several designs for a KEM have been proposed. The more common generic constructions have been analysed by Dent [16]. It will not be necessary to re-examine them in this paper.

2.5 The security criterion for a DEM

The security criteria for a DEM are the same as those for a symmetric encryption scheme. Hence, we can talk about DEMs being IND-PA and IND-CCA2 secure. This allows us to state an important theorem of Cramer and Shoup [15].

Theorem 6 *An asymmetric encryption scheme constructed from an IND-CCA2 KEM and an IND-CCA2 DEM is itself IND-CCA2 secure.*

There is a problem, however, in creating a symmetric encryption algorithm that takes arbitrarily long messages and is IND-CCA2 secure. Whilst it may be reasonable to assume that the action of a block cipher is IND-CCA2 secure, a block cipher cannot handle arbitrarily long messages. Similarly, most of the standard modes of operation of a block cipher may be able to handle arbitrarily long messages, but they are not IND-CCA2 secure even if they are used with an IND-CCA2 secure block cipher.

Cramer and Shoup [15] give an IND-CCA2 construction for a DEM. It is constructed using an IND-PA symmetric encryption scheme $(\textsc{Enc}, \textsc{Dec})$ and a secure MAC algorithm MAC. In this scheme, the encryption of a message m under a key K is given by the following algorithm.

1. Split the key K into two appropriately sized keys $K = K_1||K_2$, where K_1 is a key for the symmetric encryption scheme and K_2 is a key for the MAC scheme.
2. Set $C = \textsc{Enc}_{K_1}(m)$.
3. Set $\tau = \mathrm{MAC}_{K_2}(C)$.
4. Output the ciphertext (C, τ).

Decryption of a ciphertext (C, τ) under a key K is given by:

1. Split the key K into two appropriately sized keys $K = K_1||K_2$, where K_1 is a key for the symmetric encryption scheme and K_2 is a key for the MAC scheme.
2. Check that $\tau = \mathrm{MAC}_{K_2}(C)$. If not, output $\perp$ and stop.
3. Set $m = \textsc{Dec}_{K_1}(C)$.
4. Output m.

It is important that the keys K_1 and K_2 are of a fixed pre-determined length and are not dependent on the message or its length in any way. Cramer and Shoup show this symmetric encryption scheme to be IND-CCA2 secure provided the underlying encryption scheme is IND-PA secure and the MAC algorithm is secure.

One of the reasons that this encrypt-then-MAC construction is IND-CCA2 secure is that it in fact offers more than just a confidentiality service, it offers an integrity/authentication service too. This construction was shown to provide an integrity and confidentiality service by Bellare and Namprempre [9]. However, the encrypt-then-MAC construction is relatively inefficient, requiring the long message and the long ciphertext to be processed separately. It should be noted that other authenticated encryption modes, such as the OCB mode of operation [28], can be used as a DEMs too.

Strictly speaking, a DEM is not required to provide an authentication service but providing such a service gives the DEM "plaintext awareness", i.e. it means that it is impossible for an attacker to construct any new valid ciphertext without querying an encryption oracle. This means that a decryption oracle is essentially useless to an attacker, as the only way that the attacker can produce a ciphertext that decrypts is to query the encryption oracle and, in that case, the attacker must already know what the decryption of the ciphertext. Hence, the scheme has IND-CCA2 security even though the underlying encryption scheme is only IND-PA secure. Symmetric encryption schemes that are IND-CCA2 secure but not plaintext aware are given by Desai [17] and by Halevi and Rogaway [20, 21].

3 Hybrid Signcryption Schemes With Outsider Security

A signcryption scheme is an asymmetric scheme that combines most of the advantages of an asymmetric encryption and a digital signature scheme, specifically it provides confidentiality, integrity and authentication services. It may be advantageous for a signcryption scheme to also provide a non-repudiation service, just as a digital signature service does; however, we shall see that there are inherent problems with providing this service in this setting.

Signcryption was first studied by Zheng [32] in 1997. In his original conception of a signcryption scheme, the computational cost of the scheme had to be lower than the computational cost of performing the encryption and signature operation separately. This definition has since been expanded to include any asymmetric scheme that provides both a confidentiality and an intergity/authentication service, including schemes that make use of an asymmetric encryption scheme and a signature scheme directly. The security of signcryption schemes that make use of separate encryption and signature algorithms are discussed in [2, 4]. A method for constructing signcryption schemes from lower level primitives is also given by An and Dodis [3].

3.1 Signcryption schemes

For our purposes a signcryption scheme will consist of five algorithms:

1. A probabilistic common key generation algorithm, $\mathcal{G}_c$. It takes as input a security parameter 1^k and return some global information (parameters) I.

2. A probabilistic sender key generation algorithm, $\mathcal{G}_s$. It takes as input the global information I and outputs a public/private key pair (pk_s, sk_s) for a party who wishes to send a signcrypted message.

3. A probabilistic receiver key generation algorithm, $\mathcal{G}_r$. It takes as input the global information I and outputs a public/private key pair (pk_r, sk_r) for a party who wishes to be able to receive signcrypted messages. Hence, a party who wishes to be able to both send and receive signcrypted messages will require two key-pairs: one for use when sending messages and one who use when receiving them.

4. A probabilistic generation-encryption algorithm, $\mathcal{E}$. It takes as input a message m from some message space $\mathcal{M}$, the private key of the sender sk_s and the public key of the receiver pk_r; and outputs a signcryption $C = \mathcal{E}(sk_s, pk_r, m)$ in some signcryption space $\mathcal{C}$.

5. A deterministic verification-decryption algorithm, $\mathcal{D}$. It takes as input a signcryption $C \in \mathcal{C}$, the public key of the sender pk_s and the private key of the receiver sk_r; and outputs either a message $m \in \mathcal{M}$ or the error symbol $\perp$. We denote this operation as $\mathcal{D}(pk_s, sk_r, C)$.

The soundness condition for a signcryption scheme states that for almost all sender key pairs (pk_s, sk_s) and receiver key pairs (pk_r, sk_r) we have that $m = \mathcal{D}(pk_s, sk_r, C)$ for almost all ciphertexts $C = \mathcal{E}(sk_s, pk_r, m)$.

This definition is essentially adapted from An [2]. Some definitions include other algorithms known as the verification algorithms. The purpose of these algorithms is to provide non-repudiation. As the definitions stand, it is possible that the verification-decryption algorithm gives message origin authentication in such a way that, whilst the receiver is convinced of the origin of the message, the origin cannot be reliably demonstrated to a third party (just as in the case of a MAC algorithm being used to provide message origin authentication). The purpose of a verification algorithm is to allow a third-party, in possession of a message and a purported signcryption C of that message, to verify that C is indeed a signcryption of m without compromising the security of the scheme. We will not consider non-repudiation in this section. For more information, the reader is referred to [26].

Hence, we require our signcryption schemes to provide a confidentiality and an authentication/integrity service. In an attempt to design a hybrid

signcryption scheme, we note that these services can be provided by symmetric encryption schemes and MAC algorithms respectively. Collectively, these services can be provided by an authenticated encryption scheme, including the 'Encrypt-then-MAC' scheme used as a DEM by Cramer and Shoup. This section will show that it is possible to construct a hybrid signcryption making use of any symmetric authenticated encryption scheme (acting as DEM) and some sort of modified key encapsulation mechanism.

3.2 Outsider security for signcryption schemes

There are two potential ways in which a signcryption scheme can be attacked: an unauthorised party (i.e. not the sender or the receiver) can attempt to break the confidentiality service and learn some information about a message from its signcryption, or an unauthorised party (i.e. not the sender) can attempt to break the integrity/authentication service and forge a valid signcryption.

We sub-divide attacks against the integrity of a signcrypted message depending on the identity of the attacker. Consider an attack in which some unauthorised party attempts to forge a signcryption. In this case, the attacker could either be a third party (i.e. a party that is not involved in any legitimate transactions) or it could be a receiver who has received several valid signcrypted messages from the sender. We use the nomenclature of An, Dodis and Rabin [4] and say that a signcryption scheme is secure against outsider attacks if it secure against attacks that break the confidentiality of the message and against forgery attacks made by third parties. If a signcryption scheme is secure from forgery attacks made by valid receivers, as well as from attacks that break the confidentiality of the message, then it is said to be secure against insider attacks. It is unclear how useful a signcryption scheme with insider security is unless the scheme also offers a non-repudiation service. Obviously, a signcryption scheme that is secure against insider attacks is also secure against outsider attacks. In this section we will only consider hybrid signcryption schemes that are secure against outsider attacks.

3.2.1 Confidentiality

The notion of confidentiality for a signcryption scheme is similar to that of an asymmetric encryption scheme (see Section 2.1). The attack model is defined in terms of a game, termed the IND-CCA2 game, played between a hypothetical challenger and a two-stage attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. For a given security parameter k:

1. The challenger generates some valid parameters I by running $\mathcal{G}_c(1^k)$; a valid sender key pair (pk_s, sk_s) by running the sender key generation

algorithm $\mathcal{G}_s(I)$; and a valid receiver key pair (pk_r, sk_r) by running the receiver key generation algorithm $\mathcal{G}_r(I)$.

2. The attacker runs $\mathcal{A}_1$ on the input (pk_r, pk_s). This algorithm outputs two equal length messages, m_0 and m_1, and some state information *state*. During its execution, $\mathcal{A}_1$ can query a generation-encryption oracle that will, if given a message $m \in \mathcal{M}$, return $\mathcal{E}(sk_s, pk_r, m)$ and a verification-decryption oracle that will, if given a signcryption $C \in \mathcal{C}$, return $\mathcal{D}(pk_s, sk_r, C)$.

3. The challenger picks a bit $b \in \{0, 1\}$ uniformly at random, and computes the challenge signcryption $C^* = \mathcal{E}(sk_s, pk_r, m_b)$.

4. The attacker runs $\mathcal{A}_2$ on the input $(C^*, state)$. The algorithm outputs a guess b' for b. During its execution, $\mathcal{A}_2$ can query a generation-encryption oracle and a verification-decryption oracle as above, but with the restriction that $\mathcal{A}_2$ is not allowed to query the verification-decryption oracle on the challenge ciphertext C^*.

The attacker wins the game if $b' = b$. The attacker's advantage is defined to be:

$$|Pr[b = b'] - 1/2| \,. \tag{4}$$

Definition 7 (IND security for a signcryption scheme) *A signcryption scheme is said to be IND-CCA2 secure if, for every polynomial-time attacker $\mathcal{A}$, the advantage that $\mathcal{A}$ has in winning the IND-CCA2 game is negligible as a function of the security parameter k.*

3.2.2 Integrity/Authentication

The notion of integrity for a signcryption scheme is similar to that of a digital signature scheme (see Section 4). The attack model is defined in terms of a game, termed the sUF-CCA2 game, played between a hypothetical challenger and an attacker $\mathcal{A}$. For a given security parameter k:

1. The challenger generates some valid parameters I by running $\mathcal{G}_c(1^k)$; a valid sender key pair (pk_s, sk_s) by running the sender key generation algorithm $\mathcal{G}_s(I)$; and a valid receiver key pair (pk_r, sk_r) by running the receiver key generation algorithm $\mathcal{G}_r(I)$.

2. The attacker runs $\mathcal{A}$ on the input (pk_s, pk_r). This algorithm outputs a possible signcryption C^*. During its execution, $\mathcal{A}$ can query a generation-encryption oracle that will, if given a message $m \in \mathcal{M}$, return $\mathcal{E}(sk_s, pk_r, m)$ and a verification-decryption oracle that will, if given a signcryption $C \in \mathcal{C}$, return $\mathcal{D}(pk_s, sk_r, C)$.

The attacker wins the game if $\mathcal{D}(pk_s, sk_r, C^*) = m \neq \perp$ and $\mathcal{A}$ never received C^* as a response from generation-encryption oracle.[4]

Definition 8 (sUF security for a signcryption scheme) *A signcryption scheme is said to be sUF-CCA2 secure if, for every polynomial-time attacker $\mathcal{A}$, the probability that $\mathcal{A}$ wins the sUF-CCA2 game is negligible as a function of the security parameter k.*

Definition 9 (Outsider security for a signcryption scheme) *A signcryption scheme is said to be outsider secure if it is both IND-CCA2 and sUF-CCA2 secure.*

It should be noted that this model for outsider security gives a lot of power to the attacker: they can arbitrarily generate/encrypt signcryptions of messages and decrypt/verify signcryptions of their choice (with the exception of the challenge ciphertext). This corresponds to a real-world situation in which the attacker can trick a valid party into encrypting any message and decrypt any "unimportant" message.

3.3 A general model for a hybrid signcryption scheme

A signcryption scheme can be formed from a "signcryption KEM" and a "signcryption DEM" in the same manner as an asymmetric encryption scheme can be formed from a standard (encryption) KEM and DEM.

Definition 10 (Signcryption KEM) *A signcryption KEM is a 5-tuple of algorithms:*

1. *A probabilistic common key generation algorithm, Gen_c. It takes as input a security parameter 1^k and return some global information (parameters) I.*

2. *A probabilistic sender key generation algorithm, Gen_s. It takes as input the global information I and outputs a public/private key pair (pk_s, sk_s) for a party who wishes to send a signcrypted message.*

3. *A probabilistic receiver key generation algorithm, Gen_r. It takes as input the global information I and outputs a public/private key pair (pk_r, sk_r) for a party who wishes to be able to receive signcrypted messages.*

[4] This is sometimes known "strong unforgeability" in order to differentiate it from "weak unforgeability", where an attacker is only deemed to have won if $\mathcal{D}(pk_s, sk_r, C^*) = m \neq \perp$ and $\mathcal{A}$ never submitted m to the generation-encryption oracle. So, with strong unforgeability, an attacker is deemed to have won if it can find a new signcryption of a message that has previously been signcrypted or if it can generate a signcryption of a new message.

4. *A probabilistic key encapsulation algorithm, Encap. It takes as input a sender's private key sk_s and a receiver's public key pk_r; and outputs a symmetric key K and an encapsulation of that key C. We denote this as $(K, C) = Encap(sk_s, pk_r)$.*

5. *A deterministic key decapsulation algorithm, Decap. It takes as input a sender's public key pk_s, a receiver's private key sk_r and an encapsulation of a key C; and outputs either a symmetric key K or the error symbol $\perp$. We denote this as $K = Decap(pk_s, sk_r, C)$.*

We require that any signcryption KEM be sound, in the sense that for almost all valid sender key-pairs (pk_s, sk_s) and almost all receiver key-pairs(pk_r, sk_r) then $K = decap(pk_s, sk_r, C)$ for almost all $(K, C) = Encap(sk_s, pk_r)$.

A signcryption DEM is essentially a symmetric authenticated encryption scheme.

Definition 11 (Signcryption DEM) *A signcryption DEM is a pair of algorithms:*

1. *A deterministic encryption algorithm,* Enc*, which takes as input a message $m \in \{0,1\}^*$ of any length and a symmetric key K of some pre-determined length, and outputs an encryption $C = \text{Enc}_K(m)$ of that message.*

2. *A deterministic decryption algorithm,* Dec*, which takes as input a ciphertext $C \in \{0,1\}^*$ of any length and a symmetric key K of some pre-determined length, and outputs either a message $m = \text{Dec}_K(C)$ or the error symbol $\perp$.*

We require that any signcryption DEM be sound, in the sense that for every key K of the correct length, $m = \text{Dec}_K(\text{Enc}_K(m))$.

We combine a signcryption KEM and a signcryption DEM to form a hybrid signcryption scheme. Again we note that this is only one way in which a hybrid signcryption scheme may be formed, other hybrid signcryption schemes can be constructed that do not fit into this KEM–DEM model.

Definition 12 (KEM–DEM hybrid signcryption scheme) *Suppose that $(Gen_c, Gen_s, Gen_r, Encap, Decap)$ is a signcryption KEM,* (Enc, Dec) *is a signcryption DEM, and that, for all security parameters k, the keys produced by the signcryption KEM are of the correct length to be used by the signcryption DEM. We may then construct a signcryption scheme $(\mathcal{G}_c, \mathcal{G}_s, \mathcal{G}_r, \mathcal{E}, \mathcal{D})$ as follows:*

- *The key generation algorithms $(\mathcal{G}_c, \mathcal{G}_s, \mathcal{G}_r)$ are given by the key generation algorithms for the signcryption KEM (Gen_c, Gen_s, Gen_r).*

- *The generation-encryption algorithm $\mathcal{E}$ for a message m, a sender private key sk_s and a receiver public key pk_r is given by:*
  1. *Set $(K, C_1) = Encap(sk_s, pk_r)$.*
  2. *Set $C_2 = \text{ENC}_K(m)$.*
  3. *Output (C_1, C_2).*
- *The verification-decryption algorithm $\mathcal{D}$ for a signcryption (C_1, C_2), a sender public key pk_s and a receiver private key sk_r is given by:*
  1. *Set $K = Decap(pk_s, sk_r, C_1)$. If $K = \perp$ then output $\perp$ and stop.*
  2. *Set $m = \text{DEC}_K(C_2)$. If $m = \perp$ then output $\perp$ and stop.*
  3. *Output m.*

This construction is a sound signcryption scheme due to the soundness of the signcryption KEM and DEM. As we shall see, construction was implicitly used by An [2] in the construction of the DHETM scheme.[5]

We note that this generic construction can only achieve outsider security at best. Any party in possession of a valid receiver key pair (pk_r, sk_r) can forge a signcryption of any message m of his choice by requesting the signcryption (C_1, C_2) of some other message $m' \neq m$, recovering the symmetric key $K = Decap(pk_s, sk_r, C_1)$, computing the false ciphertext $C_2' = \text{ENC}_K(m)$ and outputting the false signcryption (C_1, C_2').

3.4 The security criteria for a signcryption KEM

In this section we will develop independent security criteria for a signcryption KEM.

3.4.1 Confidentiality

The confidentiality condition for a signcryption KEM is very similar to that of a normal (encryption) KEM: an attacker should not be able to gain any information about the symmetric key corresponding to a challenge encapsulation even if they have access to a decapsulation oracle. The only difference between the signcryption KEM and the encryption KEM cases is that we must allow the attacker access to an encapsulation oracle in the signcryption case. In the encryption case, the attacker has access to the public key and can therefore encapsulate keys themselves.

Formally, we define the confidentiality of a signcryption KEM in terms of a game played between a hypothetical challenger and a two-stage attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. We denote this the IND-CCA2 game for a signcryption KEM. For a given security parameter k, the game is as follows:

[5] Presumably DHETM stands for 'Diffie-Hellman Encrypt-then-MAC'.

1. The challenger generates some global state information I by running $Gen_c(1^k)$, a valid sender public/private key pair (pk_s, sk_s) by running $Gen_s(I)$ and a valid receiver public/private key pair (pk_r, sk_r) by running $Gen_r(I)$.

2. The attacker runs $\mathcal{A}_1$ on the input (pk_s, pk_r). It terminates by outputting some state information *state*. During this phase the attacker can query both an encapsulation oracle, which responds by returning $(K, C) = Encap(sk_s, pk_r)$, and a decapsulation oracle on an input C, which responds by returning $K = Decap(pk_s, sk_r, C)$.

3. The challenger generates a valid encapsulation (K_0, C^*) by running $Encap(sk_s, pk_r)$. It also generates a random key K_1 of the same length as K_0. Next it chooses a bit $b \in \{0, 1\}$ uniformly at random and sets $K^* = K_b$. The challenge encapsulation is (K^*, C^*).

4. The attacker runs $\mathcal{A}_2$ on the input (K^*, C^*) and *state*. It terminates by outputting a guess b' for b. During this phase the attacker can query both an encapsulation oracle and a decapsulation oracle as above, with the exception that the decapsulation oracle cannot be queried on the challenge encapsulation C^*.

The attacker wins the game if $b = b'$. If $\mathcal{A}$ made q_e queries to the encapsulation oracle and q_d queries to the decapsulation oracle then $\mathcal{A}$'s advantage is defined to be:

$$Adv\text{-}KEM\text{-}IND(q_e, q_d) = |Pr[b = b'] - 1/2| \,. \tag{5}$$

Definition 13 *A signcryption KEM is IND-CCA2 secure if, for all polynomial-time attackers $\mathcal{A}$ making at most q_e queries to the encapsulation oracle and q_d queries to the decapsulation oracle,* Adv-KEM-IND(q_e, q_d) *is negligible as a function of the security parameter k.*

It may be noted that the KEM that is implicitly used in the DHETM construction [2] does not satisfy this notion of security as a fixed key is used to encrypt all messages that pass between a particular sender and receiver. This key can easily be discovered by an attacker making a query to the encapsulation oracle. This should not be construed to mean that the DHETM scheme does not provide confidentiality, just that it does not satisfy the security criterion required for this particular method of constructing a signcryption scheme.

3.4.2 Integrity/Authentication

Whilst it is clear that if the overall hybrid signcryption scheme is going to provide an integrity/authentication service then the KEM must satisfy some

kind of extra integrity security criterion, it is a little more difficult to see what that criterion might be.

It turns out that that extra requirement is very similar to the requirement for confidentiality. For confidentiality, we declared a signcryption KEM to be secure if it was impossible for an attacker to differentiate between a valid KEM output and a random KEM output even when the attacker has access to a encapsulation and decapsulation oracles. For integrity, we will declare a signcryption KEM to be secure if it is impossible for an attacker to differentiate between a properly functioning signcryption KEM and a signcryption KEM that is giving random but consistent outputs. This notion of security is similar to the notion of Left-or-Right security for a symmetric encryption scheme [7].

In order to model this security criterion we define an ideally functioning signcryption KEM. This ideal KEM functions in exactly the same way as the normal KEM except that all of the keys it produces are randomly generated and completely unrelated to the encapsulations it produces. Consistency is maintained via a shared list of encapsulations and their associated symmetric keys. The ideal KEM is defined as the five-tuple of state-based algorithms $(Sim.Gen_c, Gen_s, Gen_r, Sim.Encap, Sim.Decap)$, where:

- The simulated common key generation algorithm, $Sim.Gen$, both runs Gen_c on the input 1^k to generate some global information I which we will be used to construct the sender and receiver public-keys, and sets up a list $KeyList$ which is initially empty.
- The simulated encapsulation algorithm, $Sim.Encap$, takes as input the pair (sk_s, pk_r) and runs as follows:
  1. Set $(K_0, C) = Encap(sk_s, pk_r)$.
  2. If there exists a pair (K_1, C) on $KeyList$ then return (K_1, C).
  3. Otherwise, generate a random symmetric key K_1 of an appropriate length, add (K_1, C) to $KeyList$ and return (K_1, C).
- The simulated decapsulation algorithm, $Sim.Decap$, takes as input the pair (pk_s, sk_r) and a signcryption C, and runs as follows:
  1. If there exists a pair (K, C) on $KeyList$ then return (K, C).
  2. If $Decap(pk_s, sk_r, C) = \perp$ then return $\perp$.
  3. Otherwise, generate a random symmetric key K of an appropriate length, add (K, C) to $KeyList$ and return K.

Informally, a signcryption KEM is Left-or-Right (LoR) secure if no polynomial-time attacker can distinguish between an execution where it has access to the proper KEM and the execution where it has access to the ideal version of the KEM.

Formally, we define Left-or-Right (LoR) security by means of a game between a hypothetical challenger and an attacker $\mathcal{A}$. We denote this game the LoR-CCA2 game for a signcryption KEM. For a given security parameter k, the game runs as follows:

1. The challenger generates some global state information I by running $Sim.Gen_c(1^k)$, a valid sender public/private key pair (pk_s, sk_s) by running $Gen_s(I)$ and a valid receiver public/private key pair (pk_r, sk_r) by running $Gen_r(I)$. The challenger also picks a bit $b \in \{0, 1\}$ uniformly at random.

2. The attacker runs $\mathcal{A}$ on the input (pk_s, pk_r). During its execution, $\mathcal{A}$ may query an encapsulation and a decapsulation oracle. If $b = 0$ then the responses to $\mathcal{A}$'s queries are computed using an encapsulation and decapsulation algorithms in the normal way. If $b = 1$ then the responses to $\mathcal{A}$'s queries are computed using the ideal encapsulation and decapsulation algorithms. $\mathcal{A}$ terminates by outputting a guess b' for b.

$\mathcal{A}$ wins the game if $b = b'$. If $\mathcal{A}$ made q_e queries to the encapsulation oracle and q_d queries to the decapsulation oracle then $\mathcal{A}$'s advantage in winning the LoR-CCA2 game is given:

$$Adv\text{-}KEM\text{-}LoR(q_e, q_d) = |Pr[b = b'] - 1/2| \,. \tag{6}$$

Definition 14 *A signcryption KEM is LoR-CCA2 secure if, for all polynomial-time attackers $\mathcal{A}$ making at most q_e queries to the encapsulation oracle and q_d queries to the decapsulation oracle,* Adv-KEM-LoR(q_e, q_d+1) *is negligible as a function of the security parameter k.*

This may seem like an odd choice for the integrity requirement of a KEM — it would make far more sense if the security criterion involved an attacker attempting to forge a valid encapsulation pair (K, C) such that $Decap(pk_s, sk_r, C) = K \neq \perp$, or, more loosely, $Decap(pk_s, sk_r, C) = K' \neq \perp$ and K and K' are related in some way, i.e. produce a new encapsulation for which he can deduce some information about the associated symmetric key. At first glance, this definition of security looks a long way from the definition of integrity that we have used but they are in fact quite similar! Consider a signcryption KEM which is LoR secure. If an attacker is able to find a valid encapsulation C for which he knows some information about the associated symmetric key then the attacker can query the decapsulation oracle on C and recover an associated symmetric key K. If K is of the form that $\mathcal{A}$ expected then $\mathcal{A}$ could conclude that the decapsulation oracle is correct: if K is not of the form that $\mathcal{A}$ expected then $\mathcal{A}$ could conclude that the decapsulation oracle is the ideal version. Hence, if $\mathcal{A}$ can forge a

valid encapsulation in such a way that he knows some information about the associated key then $\mathcal{A}$ can win the LoR-CCA2 game. Left-or-Right security is a stronger notion of security than traditional unforgeability.

Definition 15 *A signcryption KEM is said to be outsider secure if it is both IND-CCA2 and LoR-CCA2 secure.*

3.5 The security criteria for a signcryption DEM

The security criteria for a signcryption DEM are essentially the same as those for an authenticated encryption scheme [9], although we adapt these slightly for the signcryption setting.

3.5.1 Confidentiality

The confidentiality criterion for a signcryption DEM is exactly the same as for a normal (encryption) DEM: the DEM must IND-CCA2 secure as an encryption algorithm — see Section 2.2. We let *Adv-DEM-IND*(q_e, q_d) denote the advantage that an attacker has in winning the IND-CCA2 game with at most q_e encryption oracle queries and q_d decryption oracle queries.

3.5.2 Integrity/Authentication

Again, the integrity/authentication service is more difficult to determine. Normal definitions of integrity for an authenticated encryption algorithm assume that there is only one secret randomly chosen key in use, but it should be clear that a signcryption DEM needs to be secure when many different randomly chosen chosen secret keys are in use.

We define the security of a signcryption DEM in terms of a game played between a hypothetical challenger and an attacker $\mathcal{A}$. We denote this game as the INT-CCA+ game for a signcryption DEM. For a security parameter k, the game runs as follows:

1. The challenger generates a sequence $(K_1, K_2, K_3, \ldots)$ of random symmetric keys of the correct length for use by the DEM.

2. The attacker runs $\mathcal{A}$. During its execution $\mathcal{A}$ is allowed to query an encryption oracle with any input of the form (i, m) and the oracle will respond with $\textsc{Enc}_{K_i}(m)$. Similarly it may query a decryption oracle with any input of the form (i, C) and the oracle will respond with $\textsc{Dec}_{K_i}(C)$. $\mathcal{A}$ terminates by outputting a pair (i^*, C^*).

The attacker wins the game if $\textsc{Dec}_{K_{i^*}}(C^*) \neq \perp$ and C^* was never a response of the encryption oracle queried with an input of the form (i^*, m) for some message m.

Definition 16 (INT-CCA+ security for a signcryption DEM) *A signcryption DEM is said to be INT-CCA+ secure if, for every polynomial-time attacker $\mathcal{A}$, the probability that $\mathcal{A}$ wins the INT-CCA+ game is negligible as a function of the security parameter k.*

We let *Prob-DEM-IND*(q_e, q_d) denote the probability that an attacker wins the INT-CCA+ game with at most q_e encryption oracle queries and q_d decryption oracle queries.

As this security criterion requires multiple keys to be considered, it may be more difficult to prove that a given scheme is secure in the INT-CCA+ sense. Fortunately, it can be shown that a scheme that is secure in the following one-key game is also secure in the above multi-key game. For a given security parameter k, the one-key version of the INT-CCA+ game, denoted the INT-CCA game, runs as follows.

1. The challenger generates a random symmetric key K of the correct length for use by the DEM.

2. The attacker runs $\mathcal{A}$. During its execution $\mathcal{A}$ is allowed to query an encryption oracle with any message m, and the oracle will respond with $\textsc{Enc}_K(m)$. Similarly it may query a decryption oracle with any ciphertext C, and the oracle will respond with $\textsc{Dec}_K(C)$. $\mathcal{A}$ terminates by outputting a ciphertext C^*.

The attacker wins the game if $\textsc{Dec}_K(C^*) \neq \perp$ and C^* was never a response of the encryption oracle.

Definition 17 (INT-CCA security for a signcryption DEM) *A signcryption DEM is said to be INT-CCA secure if, for every polynomial-time attacker $\mathcal{A}$, the probability that $\mathcal{A}$ wins the INT-CCA game is negligible as a function of the security parameter k.*

Lemma 18 *If a signcryption DEM is INT-CCA secure then it is INT-CCA+ secure.*

Proof We prove, as always, the contra-positive statement: that if there exists an attacker that breaks the DEM with non-negligible probability in the INT-CCA+ game then there exists an attacker that breaks the DEM with non-negligible probability in the INT-CCA game. From this we can deduce that there cannot exist an attacker that wins the INT-CCA+ game with non-negligible probability as the DEM is INT-CCA secure.

Let $\mathcal{A}$ be a polynomial-time attacker that wins the INT-CCA+ game with probability at least ϵ and makes use of at most q_K keys. We define the following attacker $\mathcal{A}'$ for the INT-CCA game:

1. Choose an integer i at random from $\{1, 2, 3, \ldots, q_K\}$.

2. Choose $q_K - 1$ random symmetric keys $K_1, K_2, \ldots, K_{i-1}, K_{i+1}, \ldots, K_{q_K}$.

3. Execute $\mathcal{A}$. If $\mathcal{A}$ queries the encryption (respectively decryption) oracle with the query (j, m) (resp. (j, C)) and $j \neq i$ then return $\textsc{Enc}_{K_j}(m)$ (resp. $\textsc{Dec}_{K_j}(C)$). If $\mathcal{A}$ queries the encryption (respectively decryption) oracle with the query (i, m) (resp. (i, C)) then we pass this request on to the encryption (resp. decryption) oracle provided by the INT-CCA game and return the results to $\mathcal{A}$.

4. $\mathcal{A}$ terminates by outputting a pair (j^*, C^*). If $j^* = i$ then output C^*. Otherwise generate a ciphertext C^* uniformly at random and output C^*.

We claim that $\mathcal{A}'$ wins the INT-CCA game with probability at least ϵ/q_K.

Let W_1 be the event that $\mathcal{A}'$ wins the INT-CCA game, let W_2 be the event that $\mathcal{A}$ wins the INT-CCA+ game and let E be the event that $j^* = i$. Clearly,

$$\begin{aligned}
Pr[W_1] &\geq Pr[E \wedge W_2] && (7)\\
&= Pr[W_2] \cdot Pr[E] \text{ (since } E \text{ and } W_2 \text{ are independent)} && (8)\\
&= \epsilon \sum_{l=1}^{q_K} Pr[j^* = i | j^* = l] Pr[j^* = l] && (9)\\
&= \epsilon \sum_{l=1}^{q_K} Pr[i = l] Pr[j^* = l] && (10)\\
&= \epsilon \sum_{l=1}^{q_K} Pr[j^* = l]/q_K && (11)\\
&= \epsilon/q_K \sum_{l=1}^{q_K} Pr[j^* = l] && (12)\\
&= \epsilon/q_K \text{ (as } j^* \text{ must equal } l \text{ for some value of } l)\,. && (13)
\end{aligned}$$

Hence, the result holds. □

The notions of INT-CCA is similar to the notion of INT-CTXT suggested by Bellare and Nanprempre [9]. The only differences between the two security models is that INT-CCA model allows the attacker to have access to a decryption oracle and the INT-CTXT model allows the attacker to have multiple attempts to guess a correct ciphertext.

Definition 19 *A signcryption DEM is said to be outsider secure if it is IND-CCA2 secure and INT-CCA+ secure.*

3.6 The security of a KEM–DEM signcryption scheme

We now show that a KEM–DEM signcryption scheme built from an outsider secure KEM and an outsider secure DEM is itself outside secure. We

do this by showing that the signcryption scheme satisfies both the confidentiality and integrity/authentication criterion separately. In both cases we will show that, if there exists an attacker that breaks the signcryption scheme with non-negligible probability then there must either exist an attacker that breaks the signcryption KEM with non-negligible probability or the signcryption DEM non-negligible probability. Hence, the security of the KEM and the DEM will guarantee the security of the signcryption scheme.

We begin by showing that a KEM–DEM signcryption scheme is confidential. This proof is essentially the same as the proof of security for a generic encryption KEM–DEM scheme given by Cramer and Shoup [15].

Theorem 20 (Confidentiality of the KEM–DEM construction) *Suppose that* $(\mathcal{G}_c, \mathcal{G}_s, \mathcal{G}_r, \mathcal{E}, \mathcal{D})$ *is a hybrid signcryption scheme constructed from a signcryption KEM* $(Gen_c, Gen_s, Gen_r, Encap, Decap)$ *and a signcryption DEM* $(\textsc{Enc}, \textsc{Dec})$.

If there exists an attacker $\mathcal{A}$ *wins the IND-CCA2 game for the hybrid signcryption scheme with advantage* ϵ*, and makes at most* q_e *generation-encryption oracle queries and* q_d *verification-decryption oracle queries, then there exists an attacker* $\mathcal{B}$ *that wins the IND-CCA2 game for the signcryption KEM with advantage Adv-KEM-IND*(q_e, q_d) *and an attacker* $\mathcal{B}'$ *that wins the IND-CCA2 game for the signcryption DEM with advantage Adv-DEM-IND*(q_e, q_d) *such that*

$$\epsilon \leq 2 \cdot \text{Adv-KEM-IND}(q_e, q_d) + \text{Adv-DEM-IND}(q_e, q_d)\,. \tag{14}$$

Corollary 21 *If a hybrid signcryption scheme is constructed from an outsider secure signcryption KEM and an outsider secure signcryption DEM then that hybrid signcryption scheme is IND-CCA2 secure.*

We are therefore left with task of proving that a KEM–DEM signcryption scheme is unforgeable. For this we will also need the following simple and well-known lemma:

Lemma 22 *Suppose that an attacker* $\mathcal{A}$ *that an attacker plays a game with a challenger in attempt to determine a bit* b *which is chosen uniformly at random from* $\{0, 1\}$*. If* $\mathcal{A}$ *outputs a guess* b' *for* b*, then*

$$|Pr[b = b'] - 1/2| = 1/2 \cdot |Pr[b' = 0|b = 0] - Pr[b' = 0|b = 1]| \tag{15}$$

Proof

$$\begin{aligned}
2 \cdot |Pr[b = b'] - 1/2| &= 2 \cdot |Pr[b = b'|b = 0]Pr[b = 0] \\
&\qquad + Pr[b = b'|b = 1]Pr[b = 1] - 1/2| \\
&= |Pr[b' = 0|b = 0] + Pr[b' = 1|b = 1] - 1| \\
&= |Pr[b' = 0|b = 0] + (1 - Pr[b' = 0|b = 1]) - 1| \\
&= |Pr[b' = 0|b = 0] - Pr[b' = 0|b = 1]|
\end{aligned}$$

□

Theorem 23 (Integrity of the KEM–DEM construction) *Suppose that* $(\mathcal{G}_c, \mathcal{G}_s, \mathcal{G}_r, \mathcal{E}, \mathcal{D})$ *is a hybrid signcryption scheme constructed from a signcryption KEM* $(Gen_c, Gen_s, Gen_r, Encap, Decap)$ *and a signcryption DEM* $(\textsc{Enc}, \textsc{Dec})$.

If there exists an attacker $\mathcal{A}$ *wins the sUF-CCA2 game for the hybrid signcryption scheme with probability* ϵ*, and makes at most* q_e *generation-encryption oracle queries and* q_d *verification-decryption oracle queries, then there exists an attacker* $\mathcal{B}$ *that wins the LoR-CCA2 game for the signcryption KEM with advantage Adv-KEM-LoR*$(q_e, q_d + 1)$ *and an attacker* $\mathcal{B}'$ *that wins the INT-CCA+ game for the signcryption DEM with advantage Prob-DEM-INT*(q_e, q_d) *such that*

$$\epsilon \leq 2 \cdot \text{Adv-KEM-LoR}(q_e, q_d + 1) + \text{Prob-DEM-INT}(q_e, q_d)\,. \tag{16}$$

Proof We prove this theorem by altering the game that the challenger and the attacker play in a series of stages, and showing that an attacker's advantage is never significantly reduced by the change of game. Finally we will show that an attacker's advantage in winning the final game is small.

Let **Game 1** be the normal game that an attacker plays against a challenger, as described in Section 3.2. Let **Game 2** be the same game except that the generation-encryption and verification-decryption oracles offered to the attacker are changed to use the ideal version of the KEM described in Section 3.4, rather than the proper KEM, and that the winning condition is also slightly altered to respect this change. Hence, in **Game 2**, querying the generation-encryption oracle with a message m returns:

1. Set $(K, C_1) = \mathit{Sim.Encap}(sk_s, pk_r)$.
2. Set $C_2 = \textsc{Enc}_K(m)$.
3. Output (C_1, C_2).

and querying the verification-decryption oracle with a ciphertext (C_1, C_2) returns:

1. Set $K = \mathit{Sim.Decap}(pk_s, sk_r, C_1)$. If $K = \perp$ then output $\perp$.
2. Set $m = \textsc{Dec}_K(C_2)$. If $m = \perp$ then output $\perp$.
3. Output m.

We alter the winning condition for the integrity game for a signcryption scheme too, declaring the attacker $\mathcal{A}$ to win if it outputs a ciphertext pair (C_1, C_2) that was never return by the generation-encryption oracle, and for which:

- $Sim.Decap(pk_s, sk_r, C_1) = K \neq \perp$, and
- $\text{DEC}_K(C_2) = m \neq \perp$.

(In other words, we change the success condition of the signcryption integrity game to respect the fact that we have been using the ideal version of the KEM, rather than the proper KEM.)

Let W_1 be the event that a given attacker $\mathcal{A}$ wins **Game 1**, and let W_2 be the event that $\mathcal{A}$ wins **Game 2**. We wish to show that $|Pr[W_1] - Pr[W_2]|$ is small. We do this by defining an algorithm $\mathcal{B}$ that makes use of $\mathcal{A}$ to win the sUF-CCA2 game for the KEM. The algorithm runs as follows.

1. Initiate an (empty) list *GenEncList*.
2. Receive (pk_s, pk_r) from the challenger.
3. Execute $\mathcal{A}$ on the input (pk_s, pk_r). $\mathcal{A}$ terminates by outputting a ciphertext (C_1^*, C_2^*). If $\mathcal{A}_1$ queries a generation-encryption oracle with a message m then respond as follows.
   (a) Query the encapsulation oracle. It returns a pair (K, C_1).
   (b) Set $C_2 = \text{ENC}_K(C_1)$.
   (c) Add (C_1, C_2) to *GenEncList*.
   (d) Return (C_1, C_2).

   If $\mathcal{A}_1$ queries a verification-decryption oracle with a ciphertext (C_1, C_2) then respond as follows.
   (a) Query the decapsulation oracle on the input C_1. It will return either a key K or the error symbol $\perp$. If it responds with $\perp$ then output $\perp$.
   (b) Set $m = \text{DEC}_K(C_2)$. If $m = \perp$ then output $\perp$.
   (c) Output m.
4. Query the verification-decryption oracle on (C_1^*, C_2^*). If the oracle responds with $\perp$, or if (C_1^*, C_2^*) is on *GenEncList*, then output 1; otherwise output 0.

By lemma 22, we know that the advantage of $\mathcal{A}$ is equal to

$$1/2 \cdot |Pr[\mathcal{B} \text{ outputs } 0 | \mathcal{B} \text{ interacted with the KEM}] - Pr[\mathcal{B} \text{ outputs } 0 | \mathcal{B} \text{ interacted with the ideal KEM}]| \quad (17)$$

and so

$$1/2 \cdot |Pr[\mathcal{B} \text{ outputs } 0 | \mathcal{B} \text{ interacted with the KEM}] - Pr[\mathcal{B} \text{ outputs } 0 | \mathcal{B} \text{ interacted with the ideal KEM}]| \leq \textit{Adv-KEM-LoR}(q_e, q_d + 1) \quad (18)$$

but

$$\begin{aligned} & 1/2 \cdot |Pr[\mathcal{B} \text{ outputs } 0 | \mathcal{B} \text{ interacted with the KEM}] \\ & \quad - Pr[\mathcal{B} \text{ outputs } 0 | \mathcal{B} \text{ interacted with the ideal KEM}]| \\ & = 1/2 \cdot |Pr[W_1] - Pr[W_2]| \,. \end{aligned} \tag{19}$$

Hence,

$$|Pr[W_1] - Pr[W_2]| \leq 2 \cdot \mathit{Adv\text{-}KEM\text{-}LoR}(q_e, q_d + 1) \tag{20}$$

which is small because the KEM is secure against Left-or-Right attacks.

In **Game 2**, however, all ciphertexts are produced using random symmetric keys that are completely disassociated with the workings of the KEM. This means that the attacker is essentially attacking the DEM. Formally, we show that if there exists an attacker $\mathcal{A}$ that has a significant advantage in winning **Game 2** then there exists an attacker $\mathcal{B}'$ that wins the INT-CCA+ game for a signcryption DEM with a significant probability. The attacker $\mathcal{B}'$ runs as follows.

1. Generate some global state information I by running $\mathit{Gen}_c(1^k)$, a valid sender public/private key pair (pk_s, sk_s) by running $\mathit{Gen}_s(I)$ and a valid receiver public/private key pair (pk_r, sk_r) by running $\mathit{Gen}_r(I)$.

2. Set a counter $i = 1$ and a list of encapsulations/keys $\mathit{KeyList}$ to be empty.

3. Run $\mathcal{A}$ on (pk_s, pk_r). If $\mathcal{A}$ requests the generation-encryption of a message m then

   (a) Set $(K, C_1) = \mathit{Encap}(sk_s, pk_r)$. (N.B. K will not be used.)

   (b) If there exists a pair (j, C_1) on $\mathit{KeyList}$ for some value of j then set $C_2 = \textsc{Enc}_{K_j}(m)$ and return (C_1, C_2).

   (c) Otherwise add (i, C_1) to $\mathit{KeyList}$.

   (d) Set $C_2 = \textsc{Enc}_{K_i}(m)$.

   (e) Increase i by one and return (C_1, C_2).

   If $\mathcal{A}$ requests the verification-decryption of a ciphertext (C_1, C_2) then

   (a) If there exists a pair (j, C_1) on $\mathit{KeyList}$ for some value of j then return $m = \textsc{Dec}_{K_j}(C_2)$.

   (b) If $\mathit{Decap}(pk_s, sk_r, C_1) = \perp$ then return $\perp$.

   (c) Otherwise add (i, C_1) to $\mathit{KeyList}$.

   (d) Set $m = \textsc{Dec}_{K_i}(C_2)$.

   (e) Increase i by one and return m.

4. $\mathcal{A}$ terminates by outputting a pair (C_1, C_2). If there exists a pair (j, C_1) on $KeyList$ for some value of j then output (j, C_2). Otherwise output (i, C_2).

It should be noted that because the sequence of keys $(K_1, K_2, \ldots)$ used in the DEM INT-CCA+ security game are random generated, the probability that $\mathcal{B}'$ wins the above game is the same as the probability that $\mathcal{A}$ wins **Game 2**. Hence,

$$Pr[W_2] \leq 1/2 + \textit{Prob-DEM-INT}(q_e, q_d)\,. \tag{21}$$

Therefore, $\mathcal{A}$'s advantage in winning the signcryption integrity game (**Game 1**) is bounded by

$$2 \cdot \textit{Adv-KEM-LoR}(q_e, q_d + 1) + \textit{Prob-DEM-INT}(q_e, q_d) \tag{22}$$

□

Corollary 24 *If a hybrid signcryption scheme is constructed from an outsider secure signcryption KEM and an outsider secure signcryption DEM then that hybrid signcryption scheme is sUF-CCA2 secure.*

Corollary 25 *If a hybrid signcryption scheme is constructed from an outsider secure signcryption KEM and an outsider secure signcryption DEM then that hybrid signcryption scheme is outsider secure.*

3.7 ECISS-KEM

We now present a simple, provably secure and very efficient example of a signcryption KEM. Recall that a signcryption KEM is defined by five algorithms: a common key generation algorithm, a sender key generation algorithm, a receiver key generation algorithm, a generation-encryption algorithm and a verification-decryption algorithm. We detail each of these in turn:

- *Common key generation algorithm.* This algorithm takes as input the security parameter 1^k and outputs a triple (G, P, q) where G is a description of a suitably large cyclic group, P is a generator for that group and q is the order of the group. This group could be either a subgroup of order q of the group of multiplication modulo a prime p, i.e. $G \leq \mathbb{Z}_p^*$, or a subgroup of an elliptic curve group. We will write this group additively.

- *Sender key generation algorithm.* This algorithm picks an integer $1 \leq s \leq q-1$ uniformly at random, sets $P_s = sP$ then outputs the public key (G, P, q, P_s) and the private key (G, P, q, s).

- *Receiver key generation algorithm.* This algorithm picks an integer $1 \leq r \leq q-1$ uniformly at random, sets $P_r = rP$ then outputs the public key (G, P, q, P_r) and the private key (G, P, q, r).

- *Encapsulation algorithm.* This algorithm works as follows:

  1. Choose an element $1 \leq t \leq q-1$ uniformly at random.
  2. Set $K = Hash(sP_r + tP)$.
  3. Set $C_1 = tP$.
  4. Output (K, C_1).

- *Decapsulation algorithm.* This algorithm works as follows.

  1. Set $K = Hash(rP_s + C_1)$.
  2. Output K.

Here we assume that *hash* is a hash function that maps elements of G to bit strings of the appropriate key length.

The security of this algorithm is closely related to that of ECIES-KEM (see, for example, [12]). Due to these similarities we shall term this KEM the ECISS-KEM (Elliptic Curve Integrated Signcryption Scheme KEM). We shall present security proofs for ECISS in the random oracle model [10], despite its flaws [14]. The security of the scheme is based on the Diffie-Hellman problems:

Definition 26 *Let G be a cyclic group with prime order q (and with the group action written additively), and let P be a generator for G. The computational Diffie–Hellman problem (CDH problem) is the problem of finding abP when given (aP, bP). We assume that a and b are chosen uniformly at random from the set $\{1, \ldots, q-1\}$.*

The decisional Diffie–Hellman problem (DDH problem) is the problem of deciding whether $cP = abP$ when given (aP, bP, cP). We assume that a and b are chosen uniformly at random from the set $\{1, \ldots, q-1\}$, and c is either equal to ab (with probability 1/2) or chosen uniformly at random from the set $\{1, \ldots, q-1\}$ (with probability 1/2). The advantage that an algorithm $\mathcal{A}$ has in solving the DDH problem is equal to

$$|Pr[\mathcal{A} \text{ correctly solves the DDH problem}] - 1/2|\,.$$

The gap Diffie–Hellman problem problem (Gap DH problem) is the problem of solving the CDH problem when there exists an efficient algorithm that solves the decisional Diffie–Hellman problem on G. In other words, the gap Diffie–Hellman problem is the problem of finding abP when given (aP, bP) and access to an oracle that returns 1 when given a triple $(\alpha P, \beta P, \alpha\beta P)$ and 0 otherwise. We assume that a and b are chosen uniformly at random from the set $\{1, \ldots, q-1\}$.

3.7.1 Proof of confidentiality

The proof that ECISS-KEM is secure against attacker's attempt to win the IND-CCA2 game is similar to the conventional proof of security for ECIES-KEM.

Theorem 27 *Suppose that, in the random oracle model, there exists an attacker* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *that makes at most*

- q_e *queries to the encapsulation oracle,*
- q_d *queries to the decapsulation oracle,*
- q_h *queries to the hash function (random) oracle;*

and which can win the IND-CCA2 game against ECISS-KEM with advantage Adv*, then there exists an algorithm that can solve the CDH problem in the groups generated by the common key generation algorithm of ECISS-KEM with probability at least*

$$\frac{1}{q_h(q_e + q_d + 1)}\left\{Adv - \frac{q_e + q_d}{q}\right\}.$$

Proof We intend to use the attacker $\mathcal{A}$ as a subroutine of a larger algorithm $\mathcal{B}$ which solves the CDH problem. This algorithm will receive as input a pair of group elements (aP, bP) from which we have to deduce abP. Before we may run $\mathcal{A}$ we need to arrange two things: what are the keys for the signcryption scheme that $\mathcal{A}$ is attacking and how do we provide $\mathcal{A}$ with access to encapsulation, decapsulation and a hash (random) oracle?

It is comparatively simple to decide upon the values of the keys. We set the sender's public key P_s to be aP and the receiver's public key P_r to be bP. This mean we do not have access to the private keys for either the sender or the receiver.

In simulating the decapsulation oracle we would normally have to be sure that its responses we are consistent with the responses of the hash oracle. So that if the attacker $\mathcal{A}$ queries the decapsulation oracle on the encapsulation C they get the same results as they would get if they queried the hash oracle on the input $bP_s + C$. We maintain this consistency by keeping a record of the encapsulation, decapsulation and hash queries on two lists: a list of encapsulations and the associated keys, *EncapList*, and a list of hash function inputs and the responses, *HashList*.

It is crucial to note that if we ever know both $(C, K) \in$ *EncapList* and the corresponding $(\alpha, K) \in$ *HashList* such that $\alpha = bP_s + C$, then can deduce $\alpha - C = abP$. We simulate the encapsulation, decapsulation and hash oracles as follows.

If the attacker, $\mathcal{A}$, queries the hash (random) oracle with the input α then:

1. Check whether there exists an entry $(\alpha, K) \in \mathit{HashList}$. If so, return K to $\mathcal{A}$.

2. Otherwise, randomly generate a bit string K of the appropriate length, add (α, K) to $\mathit{HashList}$ and return K to $\mathcal{A}$.

If $\mathcal{A}$ queries the encapsulation oracle then

1. If there exists an entry $(C, K) \in \mathit{EncapList}$ then return K to $\mathcal{A}$.

2. Choose an element $1 \leq t \leq q-1$ uniformly at random.

3. Set $C = tP$.

4. Randomly generate a bit string K of the appropriate length, add (C, K) to $\mathit{EncapList}$ and return (K, C) to $\mathcal{A}$.

If $\mathcal{A}$ queries the decapsulation oracle then

1. If there exists an entry $(C, K) \in \mathit{EncapList}$ then return K to $\mathcal{A}$.

2. Otherwise, randomly generate a bit string K of the appropriate length, add (C, K) to $\mathit{EncapList}$ and return (K, C) to $\mathcal{A}$.

Note that we do not bother to to maintain consistency between the encapsulations/decapsulations oracle responses and hash function oracle responses. This means that, at some point, we may give an inconsistent response to an oracle query. We not care about this because, if it does happen, we will already have the correct entry $\mathit{HashList}$ and $\mathit{EncapList}$ to be able to deduce abP.

Consider the following algorithm $\mathcal{B}$ for solving the CDH problem:

1. Receive the challenge (aP, bP).

2. Initialise the two lists, $\mathit{HashList}$ and $\mathit{EncapList}$.

3. Execute $\mathcal{A}_1$ on the input (aP, bP). If $\mathcal{A}_1$ queries the encapsulation, decapsulation or hash oracles then respond using the simulators described above. $\mathcal{A}_1$ terminates by outputting some state information state.

4. Generate an encapsulation (K_0, C^*) using the encapsulation algorithm described above and a random symmetric key K_1. Pick a bit $\sigma \in \{0, 1\}$ uniformly at random and set the challenge encapsulation $(K^*, C^*) = (K_\sigma, C^*)$. Note that this algorithm will fail to solve the CDH problem if the challenge encapsulation C^* generated is one for which $\mathcal{A}_1$ has already queried the decapsulation oracle, or received as a response from the encapsulation oracle.

5. Execute $\mathcal{A}_2$ on the input (K^*, C^*) and *state*. If $\mathcal{A}_2$ queries the hash, encapsulation or decapsulation oracles then respond using the simulators described above. $\mathcal{A}_2$ terminates by outputting a bit σ' (which we ignore).

6. If $EncapList \neq \emptyset$ and $HashList \neq \emptyset$ then select a entry $(C, K) \in EncapList$ and an entry $(\alpha, K) \in HashList$ uniformly at random and output $\alpha - C$ as the solution to the CDH problem. Otherwise, select a point Q uniformly at random from G and output Q as the solution to the CDH problem.

We now analyse the probability that the $\mathcal{B}$ succeeds in solving a random instance of the CDH problem.

To start with, we note that the encapsulation, decapsulation and hash function oracles perfectly simulate the environment that $\mathcal{A}$ should be running in up until the point that an inconsistent response is given, at which point we have enough information to deduce the solution to the CDH problem. We also note that, due to the random oracle model, there are only two ways in which $\mathcal{A}$ can have any kind of advantage in winning the IND-CCA2 game: either $\mathcal{A}$ queried the hash function oracle on the value $bP_s + C^*$ or $\mathcal{A}_1$ made a query to the encapsulation or decapsulation oracle that involved C^* directly.

Let E be the event that $\mathcal{A}_1$ made either a query to the encapsulation oracle and received (K, C^*) back as a response, or queried the decapsulation oracle on C^*. We have that

$$\begin{aligned} Pr[\mathcal{A} \text{ wins}] &= Pr[\mathcal{A} \text{ wins}|E]Pr[E] + Pr[\mathcal{A} \text{ wins}|\neg E]Pr[\neg E] && (23) \\ &\leq Pr[E] + Pr[\mathcal{A} \text{ wins}|\neg E]\,. && (24) \end{aligned}$$

Now, since C^* is chosen at random from G, we must have that

$$Pr[E] \leq \frac{q_e + q_d}{q} \tag{25}$$

and so

$$\frac{1}{2} + Adv - \frac{q_e + q_d}{q} \leq Pr[\mathcal{A} \text{ wins}|\neg E] \tag{26}$$

Now, let F be the event that $\mathcal{A}$ makes queries that mean $(C, K) \in EncapList$ and $(bP_s + C, K') \in HashList$. Note that if F does not occur that $\mathcal{A}$ can have no clue whether the challenge encapsulation is correct or not as it has not queried the hash function oracle on $bP_s + C^*$. Hence we have that

$$\begin{aligned} \frac{1}{2} + Adv - \frac{q_e + q_d}{q} &\leq Pr[\mathcal{A} \text{ wins}|\neg E] && (27) \\ &= Pr[\mathcal{A} \text{ wins}|F \wedge \neg E]Pr[F] \\ &\quad + Pr[\mathcal{A} \text{ wins}|\neg F \wedge \neg E]Pr[\neg F] && (28) \\ &\leq Pr[F] + Pr[\mathcal{A} \text{ wins}|\neg E \wedge \neg F] && (29) \\ &= Pr[F] + 1/2 && (30) \end{aligned}$$

and therefore

$$Pr[F] \geq Adv - \frac{q_e + q_d}{q}\,. \tag{31}$$

If F occurs then $\mathcal{B}$ has at least a $1/(q_e + q_d + 1)$ chance of picking the element of *EncapList* and a $1/q_h$ chance of picking the element of *HashList* such the two entries of related. If $\mathcal{B}$ does this then $\mathcal{B}$ correctly solves the CDH problem. Hence, $\mathcal{B}$ correctly solves the CDH problem with probability at least

$$\frac{1}{q_h(q_e + q_d + 1)}\left\{Adv - \frac{q_e + q_d}{q}\right\}.$$

□

Corollary 28 *Suppose that, in the random oracle model, there exists an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that makes at most*

- *q_e queries to the encapsulation oracle,*
- *q_d queries to the decapsulation oracle,*
- *q_h queries to the hash function (random) oracle;*

and which can win the IND-CCA2 game against ECISS-KEM with advantage Adv, then there exists an algorithm that can solve the Gap-DH problem in the groups generated by the common key generation algorithm of ECISS-KEM with probability at least $Adv - (q_e + q_d)/q$.

Proof This result is easy to see once we note that, if we reduce the security to the Gap-DH problem, we do not need to guess which two entries in $(C, K) \in$ *EncapList* and $(\alpha, K) \in$ *HashList* are related but we can check each possible pairing to see if $(aP, bP, \alpha - C)$ is a valid solution to the decisional Diffie-Hellman problem using the DDH oracle.□

3.7.2 Proof of integrity/authentication

The proof of that ECISS-KEM is Left-or-Right secure is almost exactly the same as the proof that it is IND-CCA2 secure. Again, we use the fact that an attacker $\mathcal{A}$ can only distinguish between the random responses of the random oracle and the random responses of the ideal KEM if it queries the random oracle on an input for which it knows the related encapsulation — i.e. for which it has made a suitable encapsulation or decapsulation query. If such a pair of queries of queries exist then we can use them to solve an instance of the CDH problem.

Theorem 29 *Suppose that, in the random oracle model, there exists an attacker $\mathcal{A}$ that makes at most*

- *q_e queries to the encapsulation oracle,*

- *q_d queries to the decapsulation oracle,*
- *q_h queries to the hash function (random) oracle;*

and which can win the LoR-CCA2 game against ECISS-KEM with advantage Adv, then there exists an algorithm that can solve the CDH problem in the groups generated by the common key generation algorithm of ECISS-KEM with probability at least

$$\frac{Adv}{q_h(q_e + q_d)}\,.$$

Proof Again, we use the attacker $\mathcal{A}$ as a subroutine of a larger algorithm $\mathcal{B}$ which solves the CDH problem. This algorithm will receive as input a pair of group elements (aP, bP) from which we have to deduce abP. We set the public keys of the signcryption algorithm to be the same as before, i.e. set $P_s = aP$ and $P_r = bP$, and simulate the encapsulation, decapsulation and hash function oracles in exactly the same way as in Theorem 27.

Consider the following algorithm $\mathcal{B}$ for solving the CDH problem:

1. Receive the challenge (aP, bP).
2. Initialise the two lists, *HashList* and *EncapList*.
3. Execute $\mathcal{A}$ on the input (aP, bP). If $\mathcal{A}$ queries the encapsulation, decapsulation or hash oracles then respond using the simulators described above. $\mathcal{A}$ terminates by outputting a bit σ' (which we ignore).
4. If $EncapList \neq \emptyset$ and $HashList \neq \emptyset$ then select a entry $(C, K) \in EncapList$ and an entry $(\alpha, K) \in HashList$ uniformly at random and output $\alpha - C$ as the solution to the CDH problem. Otherwise, select a point Q uniformly at random from G and output Q as the solution to the CDH problem.

Now we analyse the probability that $\mathcal{B}$ solves the CDH problem. Let F be the event that $\mathcal{A}$ makes queries that mean $(C, K) \in EncapList$ and $(bP_s + C, K') \in HashList$. Note that if F does not occur that $\mathcal{A}$ can have no clue whether it is interacting with the proper KEM or the ideal KEM, due to the random nature of the hash function. Hence,

$$\begin{aligned}
1/2 + Adv &= Pr[\mathcal{A} \text{ wins}] && (32)\\
&= Pr[\mathcal{A} \text{ wins}|F]Pr[F] + Pr[\mathcal{A} \text{ wins}|\neg F]Pr[\neg F] && (33)\\
&\leq Pr[F] + Pr[\mathcal{A} \text{ wins}|\neg F] && (34)\\
&= Pr[F] + 1/2\,. && (35)
\end{aligned}$$

Therefore, $Pr[F] \geq Adv$. However, if F occurs then $\mathcal{B}$ has at least a $1/(q_e + q_d)$ chance of picking the element of *EncapList* and a $1/q_h$ chance of picking the element of *HashList* such the two entries of related. If $\mathcal{B}$ does this then

$\mathcal{B}$ correctly solves the CDH problem. Hence, $\mathcal{B}$ correctly solves the CDH problem with probability at least

$$\frac{Adv}{q_h(q_e + q_d)}.$$

□

In a manner similar to before, we can also show the following reduction.

Corollary 30 *Suppose that, in the random oracle model, there exists an attacker $\mathcal{A}$ that makes at most*

- *q_e queries to the encapsulation oracle,*
- *q_d queries to the decapsulation oracle,*
- *q_h queries to the hash function (random) oracle;*

and which can win the LoR-CCA2 game against ECISS-KEM with advantage Adv, then there exists an algorithm that can solve the Gap-DH problem in the groups generated by the common key generation algorithm of ECISS-KEM with probability at least Adv.

3.7.3 Potential weaknesses of ECISS-KEM

It is clear that ECISS-KEM is secure in the proposed security model; however, it is always a good idea to examine the security of a scheme outside of the security model in which it has been proven secure. This indicates what kind of extra information the attacker has to have access to in order to break the scheme. One weakness of ECISS-KEM is its reliance on the security of the value $sP_r + tP = rP_s + C_1 = srP + tP$.

We can view ECISS-KEM as producing a symmetric key by hashing a shared secret srP offset by a random value tP chosen by the sender. As we can see from the security proofs, if an attacker discovers this value then they can easily recover srP and break the scheme in perpetuity. Hence, it is of the utmost importance that an implementaion of the scheme keeps the value $srP + tP$ confidential.

This potential weakness could be avoided if the offset value was not easily computable by the attacker. For example, one could have an encapsulation algorithm that worked as follows.

1. Choose an element $1 \leq t \leq q - 1$ uniformly at random.
2. Set $K = Hash(sP_r + tP_r)$.
3. Set $C_1 = tP$.
4. Output (K, C_1).

The corresponding decapsulation algorithm would work as follows.

1. Set $K = Hash(rP_s + rC_1)$.
2. Output K.

This would mean that an attacker that discovered the value $sPr + tP_r = srP + trP$ would only be able to recover the single message for which that value is used to produce the symmetric key, rather than break the scheme completely. However, because it is not easy to compute srP from $sPr + tP_r$, it is a lot more difficult to produce a proof of Left-or-Right security[6] for such a scheme: it is necessary to reduce the security of the scheme to a non-standard assumption[7]. Whether an implementor wishes to use a scheme that reduces to a trusted security assumption but has a potential weakness if the security model is invalid, or use a scheme that appears more secure but reduces to an untrusted security assumption, is a very arguable implementation issue. Some arguments about this issue have been put forward by Koblitz and Menezes [24].

3.8 Using KEMs as key establishment mechanisms

One question that has been repeatedly asked since the inception of key encapsulation mechanisms has been "Can we use an (encryption) KEM as a key agreement mechanism?" Certainly KEMs exhibit the main property that we expect an asymmetric key agreement mechanism to have: they allow to remote users to pass messages between them in such a way that both users can derive a symmetric key in a suitably secure way. The simplest form of this idea is for a sender (A) to use an encryption KEM and the public key of the receiver (B) to produce a symmetric key and an encapsulation of that key (K, C), and to send the encapsulation C to the receiver who could then recover the symmetric key by running the decapsulation algorithm using their private key. Indeed, if the KEM in question is ECIES-KEM then the resulting key agreement scheme is a standardised form of the Diffie-Hellman key agreement protocol [22]. For more information about key agreement mechanisms, the reader is referred to Boyd and Mathuria [13].

The problem with key agreement mechanisms of this form is that they do not provide any kind of origin authentication or a guarantee of freshness, i.e. there is no way that B can know that they are involved in a key agreement protocol with A rather than some malicious entity claiming to be A, nor can they be sure that the message they receive is not simply a replay of an earlier execution of the protocol.

[6] An efficient proof of IND-CCA2 security that reduces the security of the scheme to the Gap Diffie-Hellman assumption can still be produced.

[7] I.e. an assumption different from the assumptions that the DDH, CDH or Gap DH problems are hard to solve.

The advent of signcryption KEMs with outsider security removes one of these problems. If one uses a signcryption KEM in the same naive way that an encryption KEM is used above, then B can at least be assured that he is engaged in a protocol exchange with A as no other entity except B can forge encapsulations purporting to come from A. This only leaves the problem of freshness.

Generally, the problem of freshness can be solved either through the use of nonces or time-stamps. A nonce is a randomly generated number that is only ever used once for the purposes of authentication, whilst a time-stamp is a digital document that contains the date/time of its creation. A simple way of adding freshness to the naive method of key agreement we have been discussing is to send either a nonce or a time-stamp along with the encapsulation. The nonce/time-stamp must be integrally protected as it is sent; this could be achieved using a MAC computed using the newly agreed secret key. Hence, the complete key agreement mechanism using time-stamps would be:

1. A uses a signcryption KEM, along with B's public key and his own private key, to generate a symmetric key and an encapsulation of that key (K, C).

2. A uses the new key to compute a MAC τ of a time-stamp t_A, and sends C, t_A and τ to B.

3. B receives C, t_A and τ, and recovers the symmetric key K by running the decapsulation algorithm on C using A's public key and B's own private key.

4. B then checks that the time-stamp t_A is current and that the τ is a MAC of the time-stamp t_A. If either of these checks fail then B rejects the key K. Otherwise B accepts the key K.

The key agreement mechanism using nonces is similar:

1. B generates a random nonce r_B and sends this to A.

2. A uses a signcryption KEM, along with B's public key and his own private key, to generate a symmetric key and an encapsulation of that key (K, C).

3. A uses the new key to compute a MAC τ of a nonce r_B, and sends C and τ to B.

4. B receives C and τ, and recovers the symmetric key K by running the decapsulation algorithm on C using A's public key and B's own private key.

5. B then checks that τ is a MAC of the nonce r_B. If this check fails then B rejects the key K. Otherwise B accepts the key K.

Whilst these examples are very simple and suffer from several practical problems, for example the lack of interaction between A and B means that the scheme becomes weak if an attacker ever compromises a key K, it does serve to show that secure key agreement mechanisms can be constructed from KEMs, but that signcryption KEMs with outsider security should be used rather than encryption KEMs.

4 Hybrid Signature Schemes

We now examine the idea of hybrid signature schemes (or, if we wish to be precise, hybrid signature schemes that do not give message recovery). Such a scheme would be expected to provide integrity, origin authentication and non-repudiation services. We note that a MAC scheme can provide integrity and origin authentication service; but, due to the fundamental shared key nature of symmetric cryptosystems, no symmetric cryptosystem can provide a non-repudiation service. We therefore aim to build a hybrid signature scheme where the symmetric 'DEM' is a MAC scheme that provides integrity protection and origin authentication, and the asymmetric 'KEM' provides a symmetric key in a non-repudiable way.

We do not make any claims that a hybrid signature scheme based on the use of a MAC is likely to be of any practical use. Indeed, it is clear to see right from the outset that these schemes are of no practical significance whatsoever. Most standard signature schemes do not sign the message itself, but a hash of the message; hence, signature schemes do not have the same problems with long messages that encryption or signcryption schemes encounter. Moreover, there exist simple signature schemes, such as RSA-PSS [11] and Schnorr [29, 30], that are likely to be at least as computationally efficient as a signature KEM. Therefore, the overall performance of a hybrid signature scheme is likely to be poor in comparison. The only reason to examine hybrid signature schemes (besides notions of completeness) is as a first step towards producing a simple hybrid signcryption scheme with *insider* security — such a scheme would share many similarities with a hybrid signature scheme.

A signature scheme is a triple of algorithms $(\mathcal{G}, \mathcal{S}, \mathcal{V})$ where:

1. $\mathcal{G}$ is a probabilistic *key generation algorithm* that takes as input a security parameter 1^k and outputs a public/private key pair (pk, sk). The private key sk is known as the signing key, while the public key pk is known as the verification key.

2. $\mathcal{S}$ is a (possible probabilistic) *signing algorithm* that takes as input a message $m \in \{0,1\}^*$ and the private key sk, and outputs a signature

σ. We denote this process by $\sigma = \mathcal{S}(sk, m)$.

3. $\mathcal{V}$ is a deterministic *verification algorithm* that takes as input a message m, a signature σ and the public key pk, and either outputs `valid` or `invalid`. We denote the use of the verification algorithm by $\mathcal{V}(pk, m, \sigma)$.

The soundness condition for a signature scheme demands that, for almost all key pairs (pk, sk), we have that $\mathcal{V}(pk, m, \sigma) = \texttt{valid}$ whenever $\sigma = \mathcal{S}(sk, m)$.

A signature scheme aspires to be existentially unforgeable (or, to be precise, strongly existentially unforgeable). This security criterion is expressed in terms of a game played between a hypothetical challenger and an attacker $\mathcal{A}$. For a given security parameter k, this game is played as follows.

1. The challenger generates a valid key-pair (pk, sk) by running the key generation algorithm $\mathcal{G}(1^k)$.

2. The attacker runs $\mathcal{A}$ on the input pk. During its execution $\mathcal{A}$ can query a signature oracle that will, when given a message m, output a signature $\sigma = \mathcal{S}(sk, m)$. $\mathcal{A}$ terminates by outputting a pair (m^*, σ^*).

The attacker wins the game if $\mathcal{V}(pk, m^*, \sigma^*) = \texttt{valid}$ and the signature oracle never responded to a query on m^* by outputting the signature σ^*.

Definition 31 (Existential unforgery of a signature scheme) *A signature scheme is said to be existentially unforgeable if, for all polynomial attackers $\mathcal{A}$, the probability that $\mathcal{A}$ wins the existential unforgeability game is negligible as a function of the security parameter k.*

On the impossibility of building a hybrid signature scheme

Let us consider trying to build a hybrid signature scheme out an asymmetric signature KEM $(Gen, Encap, Decap)$ and a symmetric signature DEM $(\textsc{Enc}, \textsc{Dec})$. We wish to consider the idea of hybrid signature scheme in a very general setting. Hence, we will allow Enc to be any deterministic polynomial-time algorithm that takes as input a symmetric key K and a message m and outputs a "ciphertext" C_2, and Dec to be any deterministic polynomial-time algorithm that takes as input a symmetric key K, a message m and a ciphertext C_2, and outputs either `valid` or `invalid`. For consistency, we require that

$$\textsc{Dec}_K(m, C_2) = \texttt{valid} \quad \text{whenever} \quad C_2 = \textsc{Enc}_K(m)\,. \tag{36}$$

It may be helpful to think of Enc as a MAC algorithm that outputs the MAC tag C_2 for the message m, and Dec as the algorithm that recomputes the MAC for a message m and compares it to the submitted tag C_2.

Based on earlier constructions it is logical to start with a KEM where the encapsulation algorithm takes the secret key sk as input, and outputs a symmetric key K and encapsulation C_1 of that key; and the decapsulation algorithm takes as input an encapsulation C_1 and the public key pk, and outputs a symmetric key K. Of course, we must require that $K = \mathit{Decap}(pk, C_1)$ whenever $(K, C_1) = \mathit{Encap}(sk)$ and (pk, sk) is a valid key pair.

A hybrid signature scheme of the following form could then be constructed.

- The key generation algorithm $\mathcal{G}$ is given by the KEM key generation algorithm Gen.
- To sign a message m using a secret key sk, the following steps are performed.
  1. Set $(K, C_1) = \mathit{Encap}(sk)$.
  2. Set $C_2 = \textsc{Enc}_K(m)$.
  3. Output (C_1, C_2).
- To verify that (C_1, C_2) is a signature for a message m using a public key pk, the following step are performed.
  1. Set $K = \mathit{Decap}(pk, C_1)$.
  2. Output $\textsc{Dec}_K(m, C_2)$.

It is easy to see that such a construction will never give a secure signature scheme. If an attacker knows that (C_1, C_2) is a valid signature for a (possibly unknown) message, then the attacker can recompute the symmetric key K used in that signature by computing $\mathit{Decap}(pk, C_1)$, and compute $C_2' = \textsc{Enc}_K(m)$ for any message m. The pair (C_1, C_2') is then a valid signature for the message m.

This attack works because the symmetric key the KEM produces is independent of the message with which it is being used. Hence, an attacker can always use the decapsulation algorithm (which is public) to recover a symmetric key K that is valid for use with all messages. Therefore, if we are going to produce a secure hybrid signature scheme, we must produce the symmetric key in a manner that depends upon the message that it is going to be used to sign. In other words, we must allow the encapsulation and decapsulation algorithm to take the message as an extra input. This, to some extent, violates the separation principle that we are trying to apply: the principle that the KEM and DEM are completely independent. We must be careful with this approach, Granboulan [19] has already shown that problems may arise if the KEM and DEM are not suitably independent. However, in this case the situation is unavoidable.

Despite this desire to keep the roles of the KEM and DEM as separate as possible, we will model the encapsulation and decapsulation algorithms as taking the whole message as an input. Therefore, the encapsulation algorithm takes a message m and a private key sk as input, and outputs a symmetric key K and an encapsulation C_2 of that key. The decapsulation algorithm takes an encapsulation C_2, a message m and the public key pk as input, and outputs a symmetric key K. Given a signature KEM of this form, the following hybrid signature scheme could be constructed.

- The key generation algorithm $\mathcal{G}$ is given by the KEM key generation algorithm *Gen*.
- To sign a message m using a secret key sk, the following steps are performed.
  1. Set $(K, C_1) = \mathit{Encap}(sk, m)$.
  2. Set $C_2 = \textsc{Enc}_K(m)$.
  3. Output (C_1, C_2).
- To verify that (C_1, C_2) is a signature for a message m using a public key pk, the following step are performed.
  1. Set $K = \mathit{Decap}(pk, m, C_1)$.
  2. Output $\textsc{Dec}_K(m, C_2)$.

In order that the simple attack used above should not apply to this new hybrid signature scheme construction, we require that it is infeasible for an attacker to be able to find an encapsulation C_1 and a message m such that $\mathit{Decap}(pk, m, C_1) \neq \perp$ without having received C_1 as a response from the signing oracle queried on the input m. If an attacker could find such a message m and and encapsulation C_1 then they could construct a valid signature (C_1, C_2) for m by computing the symmetric key $K = \mathit{Decap}(pk, m, C_1)$ and the "ciphertext" $C_2 = \textsc{Enc}_K(m)$.

However, if we assume that it is difficult for an attacker to find a message m and an encapsulation C_1 such that $\mathit{Decap}(pk, m, C_1) \neq \perp$ and C_1 was never received as a response from the signing oracle queried on the input m, then we can construct a secure signature scheme out of the KEM alone:

- The key generation algorithm $\mathcal{G}$ is given by the KEM key generation algorithm *Gen*.
- To sign a message m using a secret key sk, the following steps are performed.
  1. Set $(K, C_1) = \mathit{Encap}(sk, m)$.
  2. Output C_1

- To verify that C_1 is a signature for a message m using a public key pk, the following step are performed.

  1. Set $K = Decap(pk, m, C_1)$.
  2. If $K \neq \perp$ then output `valid`, otherwise output `invalid`.

We are therefore forced to conclude that, in order to build a KEM–DEM hybrid signature scheme, we would need to alter a secure signature scheme in such a way as to make it produce symmetric keys and then use these keys with an arbitrary DEM. The security of the scheme would rely solely on the security of the KEM and would always be less efficient than using the KEM alone.

5 Hybrid Signcryption Schemes with Insider Security

In this section we return to the problem of designing signcryption schemes, but we now attempt the more difficult task of designing a hybrid signcryption scheme with *insider* security. Just like the signcryption schemes with outsider security discussed in Section 3, signcryption schemes with insider security must protect the confidentiality of a message against attacks made by third parties (i.e. attacks made by entities who are not the sender or receiver). However, unlike signcryption schemes with outsider security, signcryption schemes with insider security must protect against forgery attacks made by any entity except the sender. In particular, this means the scheme must protect against attacks made by the receiver, who has access to the receiver's private key. This makes the task significantly more difficult.

Within this section we will use the terms "signcryption KEM", "signcryption DEM" and "hybrid signcryption scheme" to refer to a signcryption KEM with insider security, signcryption DEM with insider security and hybrid signcryption scheme with insider security (as defined in Section 5.2). We trust the reader will be able to differentiate between these components and the components used in developing hybrid signcryptions with outsider security in Section 3 by the context in which the terms are used.

5.1 Insider security for signcryption schemes

Signcryption schemes are defined in Section 3.1. A signcryption scheme must satisfy two security criteria if it is to be considered insider secure. It must protect a message's confidentiality, i.e. there must be no polynomial-time attackers who have a non-negligible advantage in winning the IND-CCA2 confidentiality game defined in Section 3.2.1. It must also protect a message's integrity against all attackers. We define this notion of security in

terms of a game, called the sIUF-CCA2 game[8], played between a hypothetical challenger and an attacker $\mathcal{A}$. For a given security parameter k:

1. The challenger generates some valid parameters I by running $\mathcal{G}_c(1^k)$; a valid sender key pair (pk_s, sk_s) by running the sender key generation algorithm $\mathcal{G}_s(I)$; and a valid receiver key pair (pk_r, sk_r) by running the receiver key generation algorithm $\mathcal{G}_r(I)$.

2. The attacker runs $\mathcal{A}$ on the input (pk_s, sk_r). This algorithm outputs a possible signcryption C^*. During its execution, $\mathcal{A}$ can query a generation-encryption oracle that will, if given a message $m \in \mathcal{M}$, return $\mathcal{E}(sk_s, pk_r, m)$. Clearly there is no need for the attacker to be given access to a verification-decryption oracle as it has access to the receiver's private key.

The attacker wins the game if $\mathcal{D}(pk_s, sk_r, C^*) = m \neq \perp$ and $\mathcal{A}$ never received C^* as a response from generation-encryption oracle[9].

Definition 32 (sIUF security for a signcryption scheme) *A signcryption scheme is said to be sIUF-CCA2 secure if, for every polynomial-time attacker $\mathcal{A}$, the probability that $\mathcal{A}$ wins is negligible as a function of the security parameter k.*

Definition 33 (Insider security for a signcryption scheme) *A signcryption scheme is said to be insider secure if it is both IND-CCA2 secure (see Section 3.2.1) and sIUF-CCA2 secure.*

5.2 A general model for a hybrid signcryption scheme

A hybrid signcryption scheme needs to provide confidentiality, integrity, and authentication services. Hence, it makes sense to try and build a hybrid signcryption scheme using an authenticated encryption scheme as the DEM; however, as we shall see, the requirements of the KEM are such that it will provide both the integrity and authentication services. Therefore, the DEM will only be needed to provide a confidentiality service.

Whilst a signcryption scheme with insider security does not automatically provide a non-repudiation service, there is a relationship between signcryption schemes with insider security and signature schemes. In both cases, there is only one entity who can successfully process a message (the signer in the signature setting and the sender in the signcryption setting). The reason that a signcryption scheme with insider security does not provide some kind

[8]Strong Insider existential UnForgeability

[9]Again, we are technically defining the notion of "strong unforgeability" as we give the attacker credit for producing a new signcryption of a previously signcrypted message. The corresponding notion of "weak unforgeability" only gives the attacker credit for producing new signcryptions of messages that have not been previously signcrypted.

of non-repudiation service is that it may be impossible for a third party to confirm that a ciphertext is a signcryption of a given message without having access to the receiver's private key. If a receiver is prepared to release his private key then a signcryption scheme with insider security will provide a non-repudiation service, but not, obviously, a confidentiality service. Hybrid signcryption schemes with non-repudiation are briefly discussed in Section 5.7.

This relationship between signcryption schemes with insider security and signcryption schemes with non-repudiation allow us to draw parallels between the problems of designing a hybrid signcryption scheme with insider security and a hybrid signature scheme. In particular, it is easy to see that the KEM of a hybrid signcryption scheme must depend upon the message on which the signcryption scheme is being used. If not, then the receiver will always be able to recover a symmetric key from an encapsulation that is valid for use with all messages, and so forge a signcryption for any message of his choice.

Within the signature setting, we made the KEM depend upon the message by allowing the message to be given as an input to the encapsulation and decapsulation algorithms. Whilst it is still possible to provide the message as an input to the encapsulation algorithm, it is not possible to provide the message as an input to the decapsulation algorithm because the receiver will not know what the message is until *after* it has been decrypted, i.e. after the decapsulation algorithm has already been executed. Hence the decapsulation algorithm must be independent of the message. We remove this seemingly paradoxical problem by proposing that a KEM contain a new, sixth algorithm *Ver*, which can be used after the message has been recovered, to indicate whether a given symmetric key is correctly associated with a given message and its encryption under the signcryption scheme. We term this a KEM's *verification algorithm*[10]. Hence, we define a signcryption KEM (for a signcryption scheme with insider security) as follows.

Definition 34 (Signcryption KEM) *A signcryption KEM is a 6-tuple of algorithms:*

1. *A probabilistic common key generation algorithm, Gen_c. It takes as input a security parameter 1^k and return some global information (parameters) I.*

2. *A probabilistic sender key generation algorithm, Gen_s. It takes as input the global information I and outputs a public/private key pair (pk_s, sk_s) for a party who wishes to send a signcrypted message.*

[10]It should be noted that a KEM's verification algorithm *Ver* is very different from the verification algorithms of a signcryption scheme with non-repudiation. The KEM's verification algorithm provides confirmation that a symmetric key, an encapsulation of that key and a message are associated with each other. The verification algorithm of a signcryption scheme is a method of providing non-repudiation evidence to third parties.

3. *A probabilistic receiver key generation algorithm, Gen_r. It takes as input the global information I and outputs a public/private key pair (pk_r, sk_r) for a party who wishes to be able to receive signcrypted messages.*

4. *A probabilistic key encapsulation algorithm, Encap. It takes as input a sender's private key sk_s, a receiver's public key pk_r and a message m; and outputs a symmetric key K and an encapsulation of that key C. We denote this as $(K, C) = Encap(sk_s, pk_r)$.*

5. *A deterministic key decapsulation algorithm, Decap. It takes as input a sender's public key pk_s, a receiver's private key sk_r and an encapsulation of a key C; and outputs either a symmetric key K or the error symbol $\perp$. We denote this as $K = Decap(pk_s, sk_r, C)$.*

6. *A deterministic verification algorithm, Ver. It takes as input a sender's public key pk_s, a receiver's private key sk_r, a message m, and an encapsulation C; and outputs either* `valid` *or* `invalid`*. Note that the verification algorithm does not need to take the symmetric key K as input as it can be easily computed from the encapsulation C using the deterministic decapsulation algorithm.*

We require that the decapsulation algorithm is sound, i.e. for almost all valid sender key-pairs (pk_s, sk_s) and almost all receiver key-pairs(pk_r, sk_r) then $K = Decap(pk_s, sk_r, C)$ for almost all $(K, C) = Encap(sk_s, pk_r)$. We also require that the verification algorithm is sound, i.e. for almost all key-pairs (pk_s, sk_s), almost all receiver key-pairs (pk_r, sk_r) and almost all $(C, K) = Encap(sk_s, pk_r, m)$ then $Ver(pk_s, sk_r, m, C) =$ `valid`.

We define a signcryption DEM (for a signcryption scheme with insider security) in exactly the same way as we defined a signcryption DEM for a signcryption scheme with outsider security in Section 3.3.

We can combine a signcryption KEM and signcryption DEM to make a signcryption scheme in the following manner.

Definition 35 (KEM-DEM hybrid signcryption scheme) *Suppose that $(Gen_c, Gen_s, Gen_r, Encap, Decap, Ver)$ is a signcryption KEM, $(\textsc{Enc}, \textsc{Dec})$ is a signcryption DEM, and that, for all security parameters k, the keys produced by the signcryption KEM are of the correct length to be used by the signcryption DEM. We may then construct a signcryption scheme $(\mathcal{G}_c, \mathcal{G}_s, \mathcal{G}_r, \mathcal{E}, \mathcal{D})$ as follows.*

- *The key generation algorithms $(\mathcal{G}_c, \mathcal{G}_s, \mathcal{G}_r)$ are given by the key generation algorithms for the signcryption KEM (Gen_c, Gen_s, Gen_r).*

- *The action of a generation-encryption algorithm $\mathcal{E}$ on a message m, a sender's private key sk_s and a receiver's public key pk_r is given by:*

1. *Set* $(K, C_1) = Encap(sk_s, pk_r, m)$.
2. *Set* $C_2 = \text{ENC}_K(m)$.
3. *Output* (C_1, C_2).

- *The action of a verification-decryption algorithm* $\mathcal{D}$ *on a signcryption* (C_1, C_2)*, a sender's public key* pk_s *and a receiver's private key* sk_r *is given by:*

  1. *Set* $K = Decap(pk_s, sk_r, C_1)$*. If* $K = \perp$ *then output* $\perp$ *and stop.*
  2. *Set* $m = \text{DEC}_K(C_2)$*. If* $m = \perp$ *then output* $\perp$ *and stop.*
  3. *If* $Ver(pk_s, sk_r, m, C_1) =$ `valid` *then output* m*. Otherwise output* $\perp$.

This construction is sound due to the soundness of the signcryption KEM and DEM.

5.3 The security criteria for a signcryption KEM

In this section we will develop independent security criteria for a signcryption KEM with insider security.

5.3.1 Confidentiality

A signcryption KEM with insider security must satisfy a similar condition to that satisfied by a signcryption KEM with outsider security, as defined in Section 3.4.1. However, we have to make two small changes. The first is to give the attacker oracle access to the verification algorithm, *Ver*. The second is a consequence of the fact that the signcryption KEM now needs to take a message as input, as well as the sender and receiver's keys.

Formally, we define the IND-CCA2 game as a game played between a hypothetical challenger and a two stage attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. For a given security parameter k, the game is played as follows.

1. The challenger generates some public parameters $I = Gen_c(1^k)$, a sender key-pair $(pk_s, sk_s) = \mathcal{G}_s(I)$ and a receiver key-pair $(pk_r, sk_r) = \mathcal{G}_r(I)$.

2. The attacker runs $\mathcal{A}_1$ on the input (pk_s, pk_r). During its execution $\mathcal{A}_1$ can query an encapsulation oracle, that will, when given a message m, return $Encap(sk_s, pk_r, m)$; a decapsulation oracle, that will, when given an encapsulation C, return $Decap(pk_s, sk_r, C)$; and a verification oracle, that will, when given an encapsulation C and a message m, return $Ver(pk_s, sk_r, m, C)$. $\mathcal{A}_1$ terminates by outputting a message m^* and some state information *state*.

3. The challenger computes the challenge signcryption as follows.
   (a) Set $(K_0, C^*) = Encap(sk_s, pk_r, m^*)$.
   (b) Randomly generate a symmetric K_1 of the same length as K_0.
   (c) Randomly generate a bit $b \in \{0, 1\}$.
   (d) Return (K_b, C^*) to the attacker.
4. The attacker executes $\mathcal{A}_2$ on the input (K^*, C^*) and *state*. During its execution $\mathcal{A}_2$ can query an encapsulation, decapsulation and verification oracle as above, with the exception that $\mathcal{A}_2$ cannot query the decapsulation oracle on the input C^*. $\mathcal{A}_2$ terminates by outputting a guess b' for b.

The attacker wins the game if $b = b'$. If an attacker makes at most q_e queries to the encapsulation oracle, q_d queries to the decapsulation oracle and q_v queries to the verification oracle then that attacker's advantage is defined to be:

$$Adv\text{-}KEM\text{-}IND(q_e, q_d, q_v) = |Pr[b = b'] - 1/2| \,. \tag{37}$$

Definition 36 *A signcryption KEM with insider security is IND-CCA2 secure if, for all polynomial-time attackers $\mathcal{A}$, that attacker's advantage in winning the IND-CCA2 game is negligible as a function of the security parameter k.*

However, along with making sure that the keys that the signcryption KEM produces are suitably random, we must now protect against the threat that a signcryption KEM leaks information about the message directly. Hence, along with a output indistinguishability criterion, a signcryption KEM must also satisfy a message indistinguishability criterion in same way as an encryption algorithm.

Formally, we define the INP-CCA2 game[11] as a game played between a hypothetical challenger and a two-stage attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. For a given security parameter k, the game is played as follows.

1. The challenger generates some public parameters $I = Gen_c(1^k)$, a sender key-pair $(pk_s, sk_s) = \mathcal{G}_s(I)$ and a receiver key-pair $(pk_r, sk_r) = \mathcal{G}_r(I)$.
2. The attacker runs $\mathcal{A}_1$ on the input (pk_s, pk_r). During its execution $\mathcal{A}_1$ can query an encapsulation oracle, that will, when given a message m, return $Encap(sk_s, pk_r, m)$; a decapsulation oracle, that will, when given an encapsulation C, return $Decap(pk_s, sk_r, C)$; and a verification oracle, that will, when given an encapsulation C and a message m, return $Ver(pk_s, sk_r, m, C)$. $\mathcal{A}_1$ terminates by outputting two messages m_0 and m_1, and some state information *state*.

[11] Here, INP stands for "input".

3. The challenger computes the challenge signcryption as follows.
   (a) Randomly generate a bit $b \in \{0, 1\}$.
   (b) Set $(K_b, C_b) = Encap(sk_s, pk_r, m_b)$.
   (c) Return C_b to the attacker.
4. The attacker executes $\mathcal{A}_2$ on the input C^* and *state*. During its execution $\mathcal{A}_2$ can query an encapsulation, decapsulation and verification oracle as above, with the exception that $\mathcal{A}_2$ cannot query the verification oracle on the inputs (m_0, C^*) or (m_1, C^*).

The attacker wins the game if $b = b'$. If an attacker makes at most q_e queries to the encapsulation oracle, q_d queries to the decapsulation oracle and q_v queries to the verification oracle then that attacker's advantage is defined to be:

$$Adv\text{-}KEM\text{-}INP(q_e, q_d, q_v) = |Pr[b = b'] - 1/2| \,. \tag{38}$$

Definition 37 *A signcryption KEM with insider security is INP-CCA2 secure if, for all polynomial-time attackers $\mathcal{A}$, that attacker's advantage in winning the INP-CCA2 game is negligible as a function of the security parameter k.*

5.3.2 Integrity/Authentication

It is clear that if an attacker, equipped with knowledge of pk_r and sk_s, can determine a KEM encapsulation C_1 and a message m such that

- $Decap(pk_s, sk_r, C_1) = K \neq \perp$,
- $Ver(pk_s, sk_r, m, C_1) = \texttt{valid}$, and
- C_1 was never the response from the KEM encapsulation oracle queried on the message m,

then that attacker can use the encapsulation C_1 to forge a new signcryption (C_1, C_2) of the message m by computing $C_2 = \textsc{Enc}_K(m)$. However, if we insist that a scheme is only secure if an attacker cannot find such a message/encapsulation pair, then we can deduce that the KEM encapsulation algorithm must be acting as a signature scheme, where the component algorithms if the signature scheme are as follows.

- Key generation is performed as follows.
  1. Set $I = Gen_c(1^k)$.
  2. Set $(pk_s, sk_s) = Gen_s(I)$.
  3. Set $(pk_r, sk_r) = Gen_r(I)$.

4. Output the private signing key $sk = (sk_s, pk_r)$ and the public verification key (pk_s, sk_r).

- The signature σ of a message m computed using a private signing key (sk_s, pk_r) is given by setting $\sigma = C$ where $(K, C) = \mathit{Encap}(sk_s, pk_r, m)$.
- A signature σ of a message m is verified using a public verification key (pk_s, sk_r) as follows.
  1. Set $K = \mathit{Decap}(pk_s, sk_r, C)$. If $K = \perp$ then output `invalid` and halt.
  2. Output $\mathit{Ver}(pk_s, sk_r, m, C)$.

Hence, any hybrid signcryption scheme with insider security must be using some kind of combination of a signature scheme and a symmetric encryption scheme directly. As a by-product we note that if the KEM is acting as a signature scheme then it is implicitly providing an integrity/authentication service for the message m; therefore, the DEM is only required to provide a confidentiality service for the message.

We define the integrity security criterion for a KEM in terms of a game played between an attacker $\mathcal{A}$ and a hypothetical challenger. This game is identical to the game that would define the security of the KEM acting as a signature scheme; however, we choose to define this game explicitly for completeness. For a given security parameter k, the game runs as follows.

1. The challenger generates some valid parameters I by running $\mathit{Gen}_c(1^k)$; a valid sender key pair (pk_s, sk_s) by running the sender key generation algorithm $\mathit{Gen}_s(I)$; and a valid receiver key pair (pk_r, sk_r) by running the receiver key generation algorithm $\mathit{Gen}_r(I)$.
2. The attacker executes $\mathcal{A}$ on the input (pk_s, sk_r). During its execution $\mathcal{A}$ can query an encapsulation oracle that will, when given a message m, output an encapsulation $(K, C) = \mathit{Encap}(sk_s, pk_r, m)$. $\mathcal{A}$ terminates by outputting a pair (m^*, C^*).

The attacker wins the game if $\mathit{Decap}(pk_s, sk_r, C^*) = K \neq \perp$, $\mathit{Ver}(pk_s, sk_r, m^*, C^*) =$ `valid`, and C^* was never the response from the encapsulation oracle queried on the message m.

Definition 38 *A signcryption KEM is INT-CCA2 secure if, for all polynomial-time attackers $\mathcal{A}$ making at most q_e queries to the encapsulation oracle, the probability that $\mathcal{A}$ wins the above integrity game,* Prob-KEM-INT(q_e)*, is negligible as a function of the security parameter k.*

The INT-CCA2 security criterion for a signcryption KEM with insider security is a much more "normal" integrity criterion than the notion of

Left-or-Right security defined for a signcryption KEM with outsider security. Left-or-Right security was introduced to make sure that the attacker couldn't produce an encapsulation where some of the bits in the symmetric key were predictable in some way, therefore making it easier to forge a DEM encryption. The idea of Left-or-Right security cannot be applied in a situation where the attacker has access to the receiver's private key and can easily check whether an encapsulation pair (K, C) given by the encapsulation oracle is correct or not.

However, it may be thought that the notion Left-or-Right security is still applicable when we consider attackers who attempt to break hybrid signcryption schemes with insider security but do not have access to the receiver's private key. This is not the case. Suppose an attacker can find an encapsulation C^* that decapsulates to give a key value $K = \mathit{Decap}(pk_s, sk_r, C^*)$ about which the attacker knows some information. The attacker still has to find a message m^* for which $\mathit{Ver}(pk_s, sk_r, m^*, C^*) = \texttt{valid}$ or the knowledge of K cannot help them break the scheme. Hence, the attacker will have had to find a valid forgery (m^*, C^*) and have broken the INT-CCA2 security of the signcryption KEM.

Definition 39 *A signcryption KEM is said to be insider secure if it is IND-CCA2, INP-CCA2 and INT-CCA2 secure.*

5.4 The security criterion for a signcryption DEM

As we have shown in Section 5.3, in a hybrid signcryption scheme with insider security the KEM provides a signature on the message. Hence, the signcryption DEM is only required to maintain the confidentiality of the message. It is therefore sufficient for the DEM to provide IND-PA security as an encryption algorithm (see Section 2.2).

We also require an oft overlooked property of a symmetric encryption scheme: we will require that the decryption algorithm is one-to-one. It is clear that the encryption algorithm of a symmetric encryption scheme is one-to-one as if $\textsc{Enc}_K(m_1) = \textsc{Enc}_K(m_2)$ then $m_1 = \textsc{Dec}_K(\textsc{Enc}_K(m_1)) = \textsc{Dec}_K(\textsc{Enc}_K(m_2)) = m_2$. However, if the range of the encryption algorithm is not equal to the entire ciphertext space, i.e. there exists valid ciphertext C that will decrypt to give a message but C is not the a possible output of the encryption algorithm, then it is possible that the decryption algorithm is not one-to-one. In such a case, and if, given a hybrid signcryption (C_1, C_2) of a message m, an attacker can find another encryption C_2' such that $\textsc{Dec}_K(C_2') = m$, where $K = \mathit{Decap}(pk_s, sk_r, C_1)$, then the attacker can forge another signcryption (C_1, C_2') for the message m.

We will term a symmetric encryption scheme with a one-to-one decryption algorithm a "one-to-one symmetric encryption scheme". Similarly, we will term a signcryption DEM consisting of a one-to-one symmetric encryp-

tion scheme, a "one-to-one signcryption DEM". All of the common symmetric encryption schemes in use are one-to-one.

Since this forgery attack produces new signcryptions for previously signcrypted messages, it is only a problem in situations that require strong unforgeability of signcryptions (see Section 5.1). However, this attack does apply to several well known signcryption schemes [5, 6, 26, 32].

5.5 The security of a KEM-DEM signcryption scheme

We will now show that a hybrid signcryption scheme made up of a secure KEM and DEM is itself secure against insider attacks. Surprisingly, we will examine the integrity/authentication properties of a hybrid signcryptions scheme first.

5.5.1 Integrity/Authentication

That a hybrid signcryption scheme is secure against insider attacks that threaten the integrity of the signcryptions is a direct result of the security of the KEM against sIUF-CCA2 attacks. The proof is very simple.

Theorem 40 (Integrity of a KEM–DEM construction) *Suppose that* $(\mathcal{G}_c, \mathcal{G}_s, \mathcal{G}_r, \mathcal{E}, \mathcal{D})$ *is a hybrid signcryption scheme constructed from a signcryption KEM* $(Gen_c, Gen_s, Gen_r, Encap, Decap, Ver)$ *and a one-to-one signcryption DEM* $(\textsc{Enc}, \textsc{Dec})$.

If there exists an attacker $\mathcal{A}$ *that wins the sIUF-CCA2 game for the hybrid signcryption scheme with advantage* ϵ*, and makes at most* q_e *queries to the generation-encryption oracle, then there exists an attacker* $\mathcal{B}$ *that wins the INT-CCA2 game for the signcryption KEM with advantage*

$$\textit{Prob-KEM-INT}(q_e) \geq \epsilon\,. \tag{39}$$

Proof Suppose that $\mathcal{A}$ is an attacker that breaks the signcryption scheme with probability ϵ. We use this to construct an algorithm $\mathcal{B}$ that breaks the sIUF-CCA2 game for the KEM with probability at least ϵ too. $\mathcal{B}$ runs as follows.

1. Receive the sender's public key pk_s and the receiver's private key sk_r from the challenger.
2. Execute $\mathcal{A}$ on the input (pk_s, sk_r). If $\mathcal{A}$ queries the generation-encryption oracle for a message m then the following steps are performed.
   (a) Query the encapsulation oracle on the input m. It will return a encapsulation-key pair (K, C_1).
   (b) Set $C_2 = \textsc{Enc}_K(m)$.
   (c) Return (C_1, C_2) to $\mathcal{A}$.

$\mathcal{A}$ terminates by outputting a signcryption (C_1^*, C_2^*).

3. Set $K^* = \mathit{Decap}(pk_s, sk_r, C_1^*)$.
4. Set $m^* = \textsc{Dec}_{K^*}(C_2^*)$.
5. Output (m^*, C_1^*).

Clearly, this algorithm perfectly simulates the environment in which $\mathcal{A}$ should be running.

Let W be the event that $\mathcal{B}$ wins the INT-CCA2 game for a signcryption KEM. Let E be the event that $\mathcal{A}$ outputs a signcryption that would win the integrity game for a signcryption scheme with insider security. Suppose that E occurs. Then $\mathcal{D}(pk_s, sk_r, C_1^*, C_2^*) = m^* \neq \perp$ and we are only required to show that C_1^* was not given by the encapsulation oracle as a response to a query on the message m^*.

If C_1^* was given as a response by the encapsulation oracle on the input query m^* then the generation-encryption oracle would have returned (C_1^*, C_2^*) to $\mathcal{A}$, as the symmetric decryption algorithm is one-to-one, and so E would not have occurred. Therefore, $\mathcal{B}$ cannot have queried the encapsulation oracle on the input m^* and received C^* as a response. Thus, (m^*, C_1^*) is a winning output for the INT-CCA2 game. Hence, the event W occurs whenever event E occurs and so

$$\epsilon = Pr[E] \leq Pr[W] = \mathit{Prob\text{-}KEM\text{-}INT}\,. \tag{40}$$

□

5.5.2 Confidentiality

The proof that a hybrid signcryption scheme provides confidentiality is only slightly more complicated than in the encryption setting [15]. We will need to make use of the following simple lemma.

Lemma 41 (Game Hopping Lemma) *If A, B and E are events in a probability space such that*

$$Pr[A|\neg E] = Pr[B|\neg E] \tag{41}$$

then

$$|Pr[A] - Pr[B]| \leq Pr[E]\,. \tag{42}$$

Proof

$$\begin{aligned} |Pr[A] - Pr[B]| &= |Pr[A|E]Pr[E] + Pr[A|\neg E]Pr[\neg E] \\ &\qquad -Pr[B|E]Pr[E] - Pr[B|\neg E]Pr[\neg E]| \\ &= |Pr[A|E]Pr[E] - Pr[B|E]Pr[E]| \\ &= |Pr[A|E] - Pr[B|E]| \cdot Pr[E] \\ &\leq Pr[E] \end{aligned}$$

□

Theorem 42 (Confidentiality of a KEM–DEM construction) *Suppose that* $(\mathcal{G}_c, \mathcal{G}_s, \mathcal{G}_r, \mathcal{E}, \mathcal{D})$ *is a hybrid signcryption scheme constructed from a signcryption KEM* $(Gen_c, Gen_s, Gen_r, Encap, Decap, Ver)$ *and a one-to-one signcryption DEM* (Enc, Dec).

If there exists an attacker $\mathcal{A}$ *that wins the IND-CCA2 game for the hybrid signcryption scheme with advantage* ϵ*, and makes at most* q_e *queries to the generation-encryption oracle and* q_d *to the verification-decryption oracle, then there exists*

- *an attacker* $\mathcal{B}$ *that wins the IND-CCA2 game for a signcryption KEM with advantage Adv-KEM-IND*(q_e, q_d, q_d)*,*
- *an attacker* $\mathcal{B}'$ *that wins the sIUF-CCA2 game for a signcryption KEM with advantage Adv-KEM-IND*$(q_e + 1)$*,*
- *an attacker* $\mathcal{B}''$ *that wins the INP-CCA2 game for a signcryption KEM with advantage Adv-KEM-INP*(q_e, q_d, q_d)*,*
- *an attacker* $\mathcal{B}'''$ *that wins the IND-PA game for a signcryption DEM with advantage Adv-DEM-IND,*

such that

$$\begin{aligned} \epsilon \;\leq\; & 2 \cdot \textit{Adv-KEM-IND}(q_e, q_d, q_d) + \textit{Adv-KEM-IND}(q_e + 1) \\ & + 2 \cdot \textit{Adv-KEM-IND}(q_e, q_d, q_d) + \textit{Adv-DEM-IND}\,. \end{aligned} \tag{43}$$

Proof We use a standard game hopping technique, which is very similar to the proof of confidentiality of a hybrid encryption scheme given by Cramer and Shoup [15]. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an attacker playing the IND-CCA2 game with the hybrid signcryption scheme. Recall that in the IND-CCA2 scheme, the attacker proposes two messages of equal length, m_0 and m_1; the challenger signcrypts one of these at random to create a challenge signcryption $(C_1^*, C_2^*) = \mathcal{E}(sk_s, pk_r, m_b)$, where b is chosen uniformly at random from $\{0, 1\}$; and the attacker has to output a guess b' for b.

We make the assumption that if $\mathcal{A}$ queries the generation-encryption oracle with a message m and receives a signcryption (C_1, C_2), then $\mathcal{A}$ never queries the verification-decryption oracle with (C_1, C_2). We justify this by noting that if $\mathcal{A}$ is an attacker than does this then there exists an equivalent attacker $\mathcal{A}'$ which does not do this and just assumes that it would receive m from the verification-decryption oracle. This fact may seem trivial but will play an important part later on.

Let **Game 1** be the game in which the attacker interacts with signcryption scheme in the manner described the IND-CCA2 game. Let **Game 2** be a similar game but instead of computing the challenge signcryption

using the key produced by the KEM, the challenger uses a randomly generated ciphertext. In other words, the challenger computes the challenge encapsulation (C_1^*, C_2^*) of a message m_b as follows.

1. Set $(K, C_1^*) = Encap(sk_s, pk_r, m_b)$.
2. Randomly generate an appropriately sized symmetric key K^*.
3. Set $C_2^* = \text{ENC}_{K^*}(m_b)$.
4. Output (C_1^*, C_2^*).

After the challenge has been issued (i.e., when the attacker is running $\mathcal{A}_2$) the challenger uses K^* to decapsulate all signcryptions of the form (C_1^*, C_2), where $C_2 \neq C_2^*$. So, if $\mathcal{A}_2$ submits a ciphertext (C_1^*, C_2) to the verification-decryption oracle then the challenger does the following.

1. Set $m = \text{DEC}_{K^*}(C_2)$.
2. If $Ver(pk_s, sk_r, m, C_1^*) = \texttt{valid}$ then output m. Otherwise output $\perp$.

Let W_1 be the probability that the attacker wins in **Game 1**. Let W_2 be the event that the attacker wins in **Game 2**. We claim that any significant difference between $Pr[W_1]$ and $Pr[W_2]$ allows us to produce an algorithm $\mathcal{B}$ that can break the security of the signcryption KEM in the IND-CCA2 game.

Consider the following algorithm $\mathcal{B}$ for breaking the KEM in the IND-CCA2 game.

1. Receive (pk_s, pk_r) from the challenger.
2. Execute $\mathcal{A}_1$ on (pk_s, pk_r). If the attacker queries the generation-encryption oracle to find the signcryption of a message m then the following steps are performed.
   (a) Query the encapsulation oracle on the input m. Receive an key/encapsulation pair (K, C_1) from the oracle.
   (b) Set $C_2 = \text{ENC}_K(m)$.
   (c) Return (C_1, C_2) to $\mathcal{A}_1$.

   If the attacker queries the verification-decryption oracle with the input (C_1, C_2) then the following steps are performed.

   (a) Query the decapsulation oracle on the input C_1. Receive a symmetric key K from the oracle. If $K = \perp$ then return $\perp$ to $\mathcal{A}_1$ and halt.
   (b) Compute $m = \text{DEC}_K(C_2)$. If $m = \perp$ then return $\perp$ to $\mathcal{A}_1$ and halt.

(c) Query the verification oracle on the input (m, C_1). If the oracle responds `valid` then return m; otherwise return $\perp$.

$\mathcal{A}_1$ terminates by outputting two equal length messages, m_0 and m_1, as well as some state information *state*.

3. We now construct the challenge signcryption for the attacker $\mathcal{A}$ as follows.

   (a) Choose a bit $b \in \{0, 1\}$ uniformly at random.

   (b) Request the challenge key/encapsulation pair from the challenger with the input m_b. The challenger returns (K^*, C_1^*).

   (c) Set $C_2^* = \text{ENC}_{K^*}(m_b)$.

4. Execute $\mathcal{A}_1$ on the input (C_1^*, C_2^*) and *state*. If the attacker queries the generation-encryption oracle then we respond as above. If the the attacker queries the verification-decryption oracle then we respond as above, except for when $\mathcal{A}_2$ queries the oracle with an input of the form (C_1^*, C_2), with $C_2 \neq C_2^*$, in which case the following steps are performed.

   (a) Set $m = \text{DEC}_{K^*}(C_2)$. If $m = \perp$ the return $\perp$ to $\mathcal{A}_2$ and halt.

   (b) Query the verification oracle on the input (m, C_1). If the oracle responds `valid` then return m; otherwise return $\perp$.

   $\mathcal{A}_2$ terminates by outputting a guess b' for b.

5. If $b = b'$ then output 0; otherwise output 1.

To analyse the success of the algorithm we use Lemma 22. Let E_1 be the event that the challenger was forced to supply the correct key K^* in the challenge. Let E_2 be the event that the challenger supplied a random key K^* in the challenge. $\mathcal{B}$'s advantage in winning the IND-CCA2 game for a signcryption KEM is given by:

$$\begin{aligned} \textit{Adv-KEM-IND} &= 1/2 \cdot |Pr[\mathcal{B} \text{ outputs } 0|E_1] \\ &\qquad - Pr[\mathcal{B} \text{ outputs } 0|E_2]| \qquad (44) \\ &= 1/2 \cdot |Pr[W_1] - Pr[W_2]| \,. \qquad (45) \end{aligned}$$

Hence, if we assume that $\mathcal{B}$'s advantage is negligible than the difference between $\mathcal{A}$'s advantage in **Game 1** and **Game 2** is negligible.

Let **Game 3** be an altered version of **Game 2** in which the challenger refuses to execute the verification-decryption algorithm on signcryptions that contain the same encapsulation as the challenge encapsulation, i.e. the challenger returns $\perp$ whenever the attacker $\mathcal{A}_2$ submits a ciphertext of the form

(C_1^*, C_2) to the verification-decryption oracle. Let E be the event that the attacker submits a signcryption to the verification-decryption oracle to which the oracle would respond with a message in **Game 2** and with $\perp$ in **Game 3**, and let W_3 be the event that the attacker $\mathcal{A}$ wins in **Game 3**. Clearly, if E does not occur then an execution of $\mathcal{A}$ is the same in both **Game 2** and **Game 3**. Hence,

$$Pr = [W_2|\neg E] = Pr[W_3|\neg E] \tag{46}$$

and so, by the game hopping lemma, we have that

$$|Pr[W_2] - Pr[W_3]| \leq Pr[E] \tag{47}$$

We claim that if $Pr[E]$ is significant then we can construct an algorithm $\mathcal{B}'$ that can break the sIUF-CCA2 game for the signcryption KEM. Consider the following algorithm $\mathcal{B}'$ for breaking the KEM in the sIUF-CCA2 game.

1. Receive (pk_s, sk_r) from the challenger.
2. Execute $\mathcal{A}_1$ on the input (pk_s, pk_r). If the attacker queries the generation-encryption oracle to find the signcryption of a message m then the following steps are performed.
   (a) Query the encapsulation oracle on the input m. Receive an key/encapsulation pair (K, C_1) from the oracle.
   (b) Set $C_2 = \text{ENC}_K(m)$.
   (c) Return (C_1, C_2) to $\mathcal{A}_1$.

   If the attacker queries the verification-decryption oracle then we execute the algorithm directly using pk_s and sk_r. $\mathcal{A}_1$ terminates by outputting two equal length message m_0 and m_1, as well as some state information *state*.
3. We now construct the challenge signcryption (C_1^*, C_2^*) as follows.
   (a) Pick a bit $b \in \{0, 1\}$ uniformly at random.
   (b) Query the encapsulation oracle on the input m_b. The oracle responds with a pair (K^*, C_1^*).
   (c) Set $C_2^* = \text{ENC}_{K^*}(m_b)$.
4. Execute $\mathcal{A}_2$ on the input (C_1^*, C_2^*) and *state*. If $\mathcal{A}_2$ queries the generation-encryption then respond as above. If $\mathcal{A}_2$ queries the verification-decryption oracle on the input (C_1, C_2) then the following steps are performed.
   (a) If $C_1 \neq C_1^*$ then return $\mathcal{D}(pk_s, sk_r, (C_1, C_2))$ to $\mathcal{A}_2$ and halt.
   (b) Check whether $\mathcal{D}(pk_s, sk_r, (C_1^*, C_2)) = \perp$. If so, return $\perp$ to $\mathcal{A}_2$ and halt.

(c) Set $m = \text{DEC}_{K^*}(C_2)$.

(d) Output (m, C_1^*) to the challenger as a forgery and stop the entire program.

$\mathcal{A}_2$ terminates by outputting a guess b' for b.

5. Randomly generate a message m and output (m, C_1^*) as a forgery.

Certainly, if the event E occurs then this algorithm will output a pair (m, C_1^*) and this will be a valid forgery unless $\mathcal{A}$ has queried the generation-encryption oracle with the message m and received the response (C_1^*, C_2'), for some value of C_2'. However, since the signcryption DEM is one-to-one, the only way that both (C_1^*, C_2) and (C_1^*, C_2') can both be a signcryption of the message m is if $C_1 = C_2'$. Hence, $\mathcal{A}$ must have queried the generation-encryption oracle with the message m and received (C_1^*, C_2) as a response. This contradicts our earlier assumption that $\mathcal{A}$ never queries the verification-decryption oracle on any signcryption it obtains as a response from the generation-encryption oracle. Therefore,

$$|Pr[W_2] - Pr[W_3]| \leq Pr[E] \leq \textit{Prob-KEM-INT}(q_e + 1)\,. \tag{48}$$

Next, we form **Game 4** by slightly altering **Game 3**. In **Game 4** we produce the encapsulation in the challenge signcryption using m_0 regardless of which message is being signcrypted. Hence, we produce the challenge signcryption as follows.

1. Generate a bit $b \in \{0, 1\}$ uniformly at random.
2. Set $(K, C_1^*) = \textit{Encap}(sk_s, pk_r, m_0)$.
3. Randomly generate an appropriately sized key K^*.
4. Set $C_2^* = \text{ENC}_{K^*}(m_b)$.
5. Return (C_1^*, C_2^*).

Let W_4 be the event that $\mathcal{A}$ wins in **Game 4**. We claim that any significant difference between $Pr[W_3]$ and $Pr[W_4]$ can be used to construct an algorithm $\mathcal{B}''$ to win the INP-CCA2 game. We can use a similar technique to the algorithm $\mathcal{B}$ that breaks the IND-CCA2 game to produce $\mathcal{B}''$. Hence, we can see that

$$|Pr[W_3] - Pr[W_4]| \leq 2 \cdot \textit{Adv-KEM-INP} \tag{49}$$

However, if the attacker $\mathcal{A}$ succeeds in winning **Game 4** then it must be attacking the DEM directly and in a passive fashion. This is because the key used to encrypt the challenge is now randomly generated and independent of the KEM, and any verification-decryption queries that $\mathcal{A}$ makes using the randomly generated key will be rejected as invalid. Hence, we can use $\mathcal{A}$ to

build an algorithm $\mathcal{B}'''$ that breaks the signcryption DEM in the IND-PA game.

Consider the following algorithm $\mathcal{B}'''$ for breaking the DEM in the IND-PA game.

1. Generate some common parameters $I = \mathcal{G}_c(1^k)$, a sender's key-pair $(pk_s, sk_s) = \mathcal{G}_s(I)$ and a receiver's key-pair $(pk_r, sk_r) = \mathcal{G}_r(I)$.

2. Execute $\mathcal{A}_1$ on the input (pk_s, pk_r). If $\mathcal{A}_1$ queries the generation-encryption oracle or the verification-decryption oracle then we respond correctly by using the keys pk_s, sk_s, pk_r and sk_r. $\mathcal{A}_1$ terminates by outputting two equal length messages m_0 and m_1, as well as some state information *state*.

3. Return (m_0, m_1) to the challenger. The challenger responds by returning a signcryption C_2^* of one of these message formed using a randomly generated symmetric key K^*. We now construct the challenge ciphertext for the attacker in the following way.

   (a) Set $(K, C_1^*) = \mathit{Encap}(sk_s, pk_r, m_0)$.

   (b) Return (C_1^*, C_2^*).

4. Execute $\mathcal{A}_2$ on the input $C_1^*, C_2^*)$ and *state*. If $\mathcal{A}_2$ queries the generation-encryption oracle or the verification-decryption oracle then we respond correctly by using the keys pk_s, sk_s, pk_r and sk_r. The only exception occurs if $\mathcal{A}_2$ queries the decapsulation oracle on a ciphertext of the form (C_1^*, C_2) when we respond with $\perp$. $\mathcal{A}_2$ terminates by outputting a bit b'.

5. Output b'.

It is clear that $\mathcal{B}'''$ exactly simulates the environment that $\mathcal{A}$ encounters in **Game 4**. Hence, $\mathcal{B}'''$ wins the IND-PA game if and only if $\mathcal{A}$ wins **Game 4** and so

$$Pr[W_4] = \mathit{Adv\text{-}DEM\text{-}IND}\,. \tag{50}$$

Thus the result holds. □

5.6 An example of a signcryption KEM

In order to provide an example of a signcryption KEM with insider security, we come full circle back to the original signcryption scheme proposed by Zheng [32]. We present the provably secure variant of Zheng's scheme proposed by Baek, Steinfeld and Zheng [5] as a KEM-DEM signcryption scheme.

- *Common key generation algorithm.* This algorithm takes as input the security parameter 1^k and outputs a triple (p, q, g) where p is a large prime, q is a large prime that divides $q - 1$ and g is an element of $\mathbb{Z}_p^*$ of order q. We will write this group multiplicatively.

- *Sender key generation algorithm.* This algorithm chooses an integer $1 \leq s \leq q$ uniformly at random, sets $P_s = g^s \bmod p$ then outputs the public key (p, q, g, P_s) and the private key (p, q, g, s).

- *Receiver key generation algorithm.* This algorithm chooses an integer $1 \leq r \leq q$ uniformly at random, sets $P_r = g^r \bmod p$ then outputs the public key (p, q, g, P_r) and the private key (p, q, g, r).

- *Encapsulation algorithm.* This algorithm works as follows.

  1. Choose an element $1 \leq t \leq q$ uniformly at random.
  2. Set $X = P_r^t \bmod p$.
  3. Set $R = \mathit{Hash}_1(m||X)$.
  4. Set $S = t/(R + s) \bmod p$.
  5. Set $K = \mathit{Hash}_2(X)$.
  6. Set $C = (R, S)$.
  7. Output (K, C).

- *Decapsulation algorithm.* This algorithm works as follows.

  1. Parse C as (R, S).
  2. Set $X = (P_s \cdot g^R)^{Sr} \bmod p$.
  3. Output $K = \mathit{Hash}_2(X)$.

- *Verification algorithm.* This algorithm works as follows.

  1. Parse C as (R, S).
  2. Set $X = (P_s \cdot g^R)^{Sr} \bmod p$.
  3. Check that $\mathit{Hash}_1(m||X) = R$. If not, output `invalid` and halt.
  4. Otherwise output `valid`.

Of course, in a real implementation of this algorithm, there is no advantage in computing X in both the decapsulation and verification algorithm. A real implementation would merely store the value of X computed by the decapsulation algorithm and use it again in the verification algorithm. Such an implementation would be functionally identical to the above algorithm and would therefore be just as secure. We choose to separate the decapsulation and verification algorithms so that they can be studied independently.

The proofs of security for this algorithm can be adapted from those contained [5].

5.7 Non-repudiation

It has been suggested that signcryption schemes with insider security are not much use unless they also offer a non-repudiation service. The problems associated with designing a signcryption scheme with non-repudiation have been extensively studied by Malone-Lee [26].

If we aim to design a useful signcryption scheme with non-repudiation then it is best if the non-repudiation algorithms are non-interactive. In such a case, the receiver of the message is able to release some extra information about a signcryption that allows third-parties to be able to verify the validity of that signcryption (i.e., that is is a signcryption of a particular message sent from the sender to the receiver). However, it is important that the release of this information does not allow the attacker to compromise future signcryptions. So, for example, whilst a receiver can prove the authenticity of a signcryption by releasing his secret key, this solution is not practical as it means that all the verifying third parties will be able to read all past and future signcryptions.

Zheng [32] proposes a solution whereby the receiver proves the authenticity of a signcryption C by releasing the associated message m and a non-interactive zero-knowledge proof that m is the message associated with C. We note that, with a hybrid signcryption scheme, all that is required to provide a non-repudiation service on the signcryption (C_1, C_2) is for the receiver to release the symmetric key K and a non-interactive zero-knowledge proof that K is the symmetric key associated with the encapsulation C_1 for the message $\textsc{Dec}_K(C_2)$. Of course, care must still be taken to ensure that the release of a key K does not enable further attacks on the signcryption scheme.

6 Hybrid Signcryption Schemes in a Multi-User Setting

Throughout this paper we have only considered the so-called "two user" setting. That is, we have only considered a signcryption scheme set up to allow two users to communicate with each other. In reality this is unlikely to occur. Signcryption schemes are likely to be used with a large number of different users, where each user in the system has there own set of sender and receiver public/private key pairs. In such a system an attacker may have a great deal more power than in the two-user setting. For example, the attacker may be able to register new public/private key pairs (either randomly generated or selected by the attacker), or the attacker may be able to corrupt legitimate users and learn their private keys.

Baek, Steinfeld and Zheng [5] proposed a formal model for signcryption which, whilst similar to the model for insider security discussed in Section 5.1

and proposed by An, Dodis and Rabin [4], allows the attacker to query a "flexible decryption-verification oracle" which executes the decryption-verification algorithm on a given signcryption with a given private signcryption key. Malone-Lee [26] goes further and proposes a model whereby the attacker can run the both generation-encryption and verification-decryption oracle for any user, but this model still may not go far enough. Neither model allows the user to corrupt entities and learn the values of their private keys. An, Dodis and Rabin also briefly consider the problem of using "black-box" signature and encryption schemes to create signcryption schemes in the multi-user setting.

Whilst it is not the intension of this paper to propose such a model, we will note that there appears to be a simple solution to the problem of adapting two-user secure hybrid signcryption schemes to the multi-user setting for *most* hybrid signcryption schemes. We note that almost all hybrid schemes in the two-user setting produce a symmetric secret key K by taking the hash of some secret value α. To adapt such a series of schemes to the multi-user setting all that would be required is to change this key derivation process so that the symmetric key K is instead formed by taking the hash of both the secret value α and the sender and receiver's public keys pk_s and pk_r.

7 Conclusions

We have shown that the KEM–DEM approach to hybrid cryptography can be used for more than just encryption. In particular, it is very useful paradigm for constructing signcryption schemes (although it is not very useful for constructing signature schemes). Perhaps the most intriguing aspect of this work is the potential for a streamlined piece of software that allows a user to choose between encryption, signcryption with outsider security and signcryption with insider security by just changing the nature of the KEM and DEM that are being used. Of course, in such a case, care would have to be taken to ensure that suitably independent public/private key pairs were used for each possible KEM/DEM pair.

Acknowledgements

The author would like to thank John Malone-Lee, Liqun Chen, Fred Piper, Bodo Möller, Yevgeniy Dodis and Stéphanie Alt for their helpful comments. The author also gratefully acknowledges the financial support of the EPSRC.

References

[1] M. Abdalla, M. Bellare, and P. Rogaway. DHAES: An encryption scheme based on the Diffie-Hellman problem. Submission to *P1363a: Standard Specifications for Public-Key Cryptography, Additional Techniques*, 2000.

[2] J. H. An. Authenticated encryption in the public-key setting: Security notions and analyses. Available from `http://eprint.iacr.org/2001/079`, 2001.

[3] J. H. An and Y. Dodis. Concealment and its applications to authenticated encryption. In E. Biham, editor, *Advances in Cryptology – Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 312–329. Springer-Verlag, 2003.

[4] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In L. Knudsen, editor, *Advances in Cryptology – Eurocrypt 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer-Verlag, 2002.

[5] J. Baek, R. Steinfeld, and Y. Zheng. Formal proofs for the security of signcryption. In D. Naccache and P. Pallier, editors, *Public Key Crytpography 2002 (PKC 2002)*, volume 2274 of *Lecture Notes in Computer Science*, pages 80–98. Springer-Verlag, 2002.

[6] F. Bao and R. H. Deng. A signcryption scheme with signature directly verifiable by public key. In H. Imai and Y. Zheng, editors, *Public Key Cryptography 1998 (PKC '98)*, volume 1431 of *Lecture Notes in Computer Science*, pages 55–59. Springer-Verlag, 1998.

[7] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings of the 38th Symposium on Foundations of Computer Science*, IEEE, 1997.

[8] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology – Crypto '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45. Springer-Verlag, 1998.

[9] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *Advances in Cryptology – Asiacrypt 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer-Verlag, 2000.

[10] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. of the First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[11] M. Bellare and P. Rogaway. The exact security of digital signatures — how to sign with RSA and Rabin. In U. Maurer, editor, *Advances in Cryptology – Eurocrypt '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer-Verlag, 1996.

[12] I. Blake, G. Seroussi, and N. Smart, editors. *Elliptic Curves in Cryptography II: Further Topics*. Cambridge Univeristy Press, 2004.

[13] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Springer-Verlag, 2003.

[14] R. Canetti, O. Goldreich, and S. Halvei. The random oracle model, revisited. In *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing*, pages 209–218, 1998.

[15] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2004.

[16] A. W. Dent. A designer's guide to KEMs. In K. G. Paterson, editor, *9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151. Springer-Verlag, 2003.

[17] A. Desai. New paradigms for constructing symmetric encryption schemes secure against chosen-ciphertext attack. In M. Bellare, editor, *Advances in Cryptology – Crypto 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 394–412. Springer-Verlag, 2000.

[18] Y. Dodis, M. J. Freedman, S. Jarecki, and S. Walfish. Optimal signcryption from any trapdoor permutation. Available from `http://eprint.iacr.org/2004/020/`, 2004.

[19] L. Granboulan. RSA hybrid encryptoin schemes. Available from `http://eprint.iacr.org/2001/110/`, 2001.

[20] S. Halevi and P. Rogaway. A tweakable enciphering mode. In D. Boneh, editor, *Advances in Cryptology – Crypto 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 482–499. Springer-Verlag, 2003.

[21] S. Halevi and P. Rogaway. A parallelizable enciphering mode. In T. Okamoto, editor, *Topics in Cryptography – CT-RSA 2004*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304. Springer-Verlag, 2004.

[22] International Organization for Standardization. *ISO/IEC 11770–3, Information technology — Security techniques — Key Management — Part 3: Mechanisms using asymmetric techniques*, 1999.

[23] International Organization for Standardization. *ISO/IEC 18033–1, Information technology — Security techniques — Encryption Algorithms — Part 1: General*, 2003.

[24] N. Koblitz and A. J. Menezes. Another look at "provable security". Available from `http://eprint.iacr.org/2004/152/`, 2004.

[25] S. Lucks. A variant of the Cramer-Shoup cryptosystem for groups of unknown order. In Y. Zheng, editor, *Advances in Cryptology – Asiacrypt 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 27–45. Springer-Verlag, 2002.

[26] J. Malone-Lee. Signcryption with non-interactive non-repudiation. Technical Report CSTR-02-004, Department of Computer Science, University of Bristol, May 2004.

[27] T. Okamoto, S. Uchiyama, and E. Fujisaki. EPOC: Efficient probabilistic prublic-key encryption. Submission to *P1363a: Standard Specifications for Public-Key Cryptography, Additional Techniques*, 1999.

[28] P. Rogaway, M. Bellare, J. Black, and T. Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In *Proceedings of the Eighth ACM Conference on Computer and Communications Security (CCS-8)*, pages 196–205. ACM Press, 2001.

[29] C. P. Schnorr. Efficient signature generation for smart cards. In G. Brassard, editor, *Advances in Cryptology – Crypto '89*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252. Springer-Verlag, 1989.

[30] C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[31] V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In B. Preneel, editor, *Advances in Cryptology – Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 275–288. Springer-Verlag, 2000.

[32] Y. Zheng. Digital signcryption or how to achieve cost(signature & encryption) << cost(signature) + cost(encryption). In B. Kaliski, editor, *Advances in Cryptology – Crypto '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer-Verlag, 1997.