Hybrid Signcryption Schemes with Insider Security (Extended Abstract)

Alexander W. Dent

Royal Holloway, University of London
Egham Hill, Egham, Surrey, TW20 0EX, U.K.
a.dent@rhul.ac.uk
http://www.isg.rhul.ac.uk/~alex/

Abstract. The question of constructing a hybrid signcryption scheme with outside security was considered by Dent [7]. That paper also demonstrated that the basic hybrid construction formalised by Cramer and Shoup [5, 9] is incapable of producing a signcryption scheme with insider security. This paper provides a paradigm for constructing signcryption schemes with insider security based on the ideas of hybrid cryptography.

1 Introduction

Hybrid cryptography is concerned with the combination of keyed symmetric and asymmetric schemes in order to produce schemes that are more advantageous than those constructed using "pure" symmetric or asymmetric techniques alone. Typically, this takes the form of an asymmetric cryptosystem making use of a generic keyed symmetric cryptosystem with certain security properties as a subroutine. This enables the construction of asymmetric schemes in which some of the computational load is taken by the more efficient symmetric cryptosystems without compromising the security of the overall cryptosystem. Traditionally, hybrid cryptography is used to create asymmetric encryption schemes where the actual encryption of the message is provided by a symmetric encryption scheme (for example, AES in CBC mode) under a randomly generated symmetric key. The asymmetric encryption scheme is then used to encrypt this randomly generated symmetric key. This allows the asymmetric encryption scheme to handle long messages, a problem with some "pure" asymmetric encryption schemes.

Cramer and Shoup [5, 9] proposed a paradigm whereby the asymmetric and symmetric parts of the cryptosystem are formally separated into an asymmetric KEM and a symmetric DEM. The authors proposed separate security criteria for the KEM and the DEM that would, if fulfilled, guarantee that the overall encryption scheme was secure. Dent [7] extended this paradigm to the signcryption setting by proposing new security criteria for the KEM and the DEM, although the model only covers the case where the signcryption scheme was attacked by third parties (known as *outsiders* by An *et al.* [2]).

This paper extends earlier work by proposing a hybrid paradigm for signcryption schemes secure against attacks made by *insiders*, i.e. the resultant schemes

should be secure against attacks against the confidentiality of the message made by any third party and from forgery attacks made by any person except the sender. We also note the infeasibility of producing efficient hybrid signature schemes.

2 KEMs, DEMs and the Impossibility of Hybrid Signature Schemes

A KEM–DEM encryption scheme is composed of two parts: an asymmetric *key encapsulation mechanism* (KEM) and a symmetric *data encapsulation mechanism* (DEM). To encrypt a message m, the KEM is first used, with the public key, to produce both a symmetric key K and an asymmetric encryption (or "encapsulation") of that key C_1. The message m is then encrypted symmetrically using the DEM and the randomly generated symmetric key K to give a ciphertext C_2. The encryption ciphertext is the pair (C_1, C_2). An encryption ciphertext (C_1, C_2) is decrypted by first decapsulating C_1 with the KEM and the private key to recover the symmetric key K, and then using the DEM and the symmetric key K to recover the message m from C_2.

As a step towards building a hybrid signcryption scheme, we consider the problem of building a hybrid signature scheme. A signature scheme needs to provide integrity, data origin authentication and non-repudiation services. Since a symmetric MAC scheme can provide both a integrity and an origin authentication service, we may have some hope that we can build a hybrid signature scheme[1,2].

Naively we may try and build a hybrid signature scheme (that uses a public verification key pk and a private signing key sk) for a message m as follows. To sign a message m:

1. The KEM is executed on the private key sk to produce a symmetric key K and an encapsulation of that key C_1.
2. The DEM is executed on the message m and the symmetric key K, and produces a cryptographic check value C_2.

The signature is the pair (C_1, C_2). To verify such a signature:

1. The KEM is executed using the public key pk and the first part of the signature C_1, and outputs either a symmetric key K or the error symbol $\perp$.

[1] Of course, no symmetric scheme can provide a non-repudiation service without the use of a trusted third party. Hence, the KEM must act in such a way as to make sure that the overall scheme provides a non-repudiation service.

[2] It is unlikely that a hybrid signature scheme is likely to be of much practical use. It is likely that any KEM, due to its asymmetric nature, is likely to contain at least one "slow" operation (such as modular exponentiation or scalar multiplication of an elliptic curve point). Since there exist fast signature algorithm that only makes use of one "slow" operation, such as RSA-PSS [4] and Schnorr [8], any hybrid signature scheme is likely to be slower than its "pure" counterpart.

If the KEM outputs $\perp$, then the verification algorithm outputs *invalid* and terminates.
2. The DEM is executed using the message m, the symmetric key K and the second part of the ciphertext C_2. The DEM outputs either *valid* or *invalid*. The verification algorithm outputs either *valid* or *invalid* depending on the DEM's output.

It is easy to see that no hybrid signature scheme of this form can ever be secure. An attacker can always forge a signature for any message m by requesting the signature (C_1, C_2) of a message m' from the signer, recovering the symmetric key K used to create the signature (from C_1 and pk), and creating a new cryptographic check value C_2' by executing the DEM on the message m using the symmetric key K. The pair (C_1, C_2') is a valid signature for the message m. If we are to avoid this problem then we are forced to alter the KEM–DEM paradigm.

Definition 1. *A signature KEM is a triple* $(Gen, Encap, Decap)$ *where*

- *Gen is a probabilistic algorithm that take a security parameter* 1^k *and outputs a public/private key pair* (pk, sk)*, where* sk *is a private signing key and* pk *is the corresponding public verification key.*
- *Encap is a probabilistic key encapsulation algorithm that takes as input a private key* sk *and a message* m*, and outputs a symmetric key* K *and an encapsulation of that key* C_1*.*
- *Decap is a deterministic key decapsulation algorithm that takes as input a public key* pk*, a message* m *and an encapsulation* C_1*, and outputs either a symmetric key* K *or the error symbol* $\perp$*.*

We require that a signature KEM is sound, i.e. if (pk, sk) *is a public/private key pair,* m *is a message, and* $(K, C_1) = Encap(sk, m)$ *then* $K = Decap(pk, m, C_1)$*.*

Thus, the KEM produces symmetric keys that depend on the message being signed, as well as the public key.

Definition 2. *A signature DEM is a pair* (Compute, Check) *where*

- Compute *is a deterministic algorithm that takes as input a message* m *and a symmetric key* K*, and outputs a cryptographic check value* C_2*.*
- Check *is a deterministic algorithm that takes as input a message* m*, a symmetric key* K *and a check value* C_2 *and outputs* valid *if* $\textsc{Compute}(m, K) = C_2$ *and* invalid *otherwise.*

We construct a signing algorithm for a message m as follows:

1. Set $(K, C_1) = Encap(sk, m)$.
2. Set $C_2 = \textsc{Compute}(m, K)$.
3. Output (C_1, C_2).

The corresponding verification algorithm for a message m and a signature (C_1, C_2) is:

1. Set $K = Decap(pk, m, C_1)$. If $K = \perp$ then output *invalid* and terminate the algorithm.
2. Output $\textsc{Check}(m, K, C_2)$.

Now, to defeat the earlier simple attack, we require that no attacker be able to compute an encapsulation C_1 for a symmetric key K used to encrypt a message m except by querying a signature oracle. If an attacker can find such an encapsulation for a message m, then they can recover K using the public verification algorithm and compute $C_2 = \textsc{Compute}_K(m)$ using the DEM. The pair (C_1, C_2) would be a valid signature for the message m. However, if we insist upon the KEM having this security property then we can construct a signature scheme from the KEM alone as follows. To sign a message m under a private key sk:

1. Set $(K, C_1) = Encap(sk, m)$.
2. Output C_1.

To verify a signature C_1 of a message m under a public key pk:

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

Hence, whenever we have a secure hybrid signature scheme, we can construct a more efficient secure signature scheme by simply removing the DEM.

It is not surprising that we cannot construct an efficient secure hybrid signature scheme. Since the DEM can only provide integrity and data origin authentication services, the responsibility of providing the non-repudiation property falls to the KEM. However, a KEM that is providing a non-repudiation service also provides integrity and data origin authentication services.

3 Insider Secure Signcryption Schemes

The notion of signcryption was first introduced by Zheng [10]. For our purposes a signcryption scheme will consist of five algorithms:

1. A probabilistic polynomial-time common key generation algorithm, $\mathcal{G}_c$. It takes as input a security parameter 1^k and returns some global information (parameters) I.
2. A probabilistic polynomial-time 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 signcrypted messages.
3. A probabilistic polynomial-time 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 for use when receiving them.

4. A probabilistic polynomial-time 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 ciphertext space $\mathcal{C}$.
5. A deterministic polynomial-time 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 $m = \mathcal{D}(pk_s, sk_r, C)$ or the error symbol $\perp$.

We require that any signcryption scheme is *sound*, i.e. that for almost all sender key pairs (pk_s, sk_s) and receiver key pairs (pk_r, sk_r) we have $m = \mathcal{D}(pk_s, sk_r, C)$ for almost all ciphertexts $C = \mathcal{E}(sk_s, pk_r, m)$. This definition of a signcryption scheme is essentially adapted from An [1].

We take our security notion for a signcryption scheme from An, Dodis and Rabin [2]. When we consider attacks against a signcryption scheme we have to consider two different types of attack. We have to consider attacks against the confidentiality of the scheme made by any third party (i.e. any party who is not the sender or the receiver); and we have to consider attacks made against the integrity of the scheme made by any party except the sender. This is known as *insider* security[3].

Both attacks against the confidentiality and attacks against the integrity are described in terms of a game played between an attacker and a hypothetical challenger. In each case the system is secure if the attacker's success probability or advantage is negligible as a function of the security parameter.

Definition 3. *A function f is negligible if, for all polynomials p, there exists an integer N_p such that $|f(x)| \leq 1/|p(x)|$ for all $x \geq N_p$.*

Confidentiality

The notion of confidentiality for a signcryption scheme is similar to that of an asymmetric encryption scheme. The attack model is defined in terms of 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:

1. The challenger generates some global information I by running the common key generation algorithm $\mathcal{G}_c(1^k)$; 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 weaker notion of *outsider* security protects against attacks against the confidentiality or integrity made by any third party, but does not protect against attacks against the integrity made by the receiver.

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:

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

Definition 4 (IND-CCA security). *A signcryption scheme is said to IND-CCA secure if, for all polynomial polynomial-time attackers $\mathcal{A}$, the advantage that $\mathcal{A}$ has in winning the above game is negligible as a function of the security parameter k.*

Integrity/Authenticity

The notion of integrity for a signcryption scheme is similar to that of a digital signature scheme. The attack model is defined in terms of a game played between a hypothetical challenger and an attacker $\mathcal{A}$. For a given security parameter k:

1. The challenger generates some global information I by running the common key generation algorithm $\mathcal{G}_c(1^k)$; 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, 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 $\mathcal{E}(sk_s, pk_r, m)$.

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 5 (INT-CCA security). *A signcryption scheme is said to be INT-CCA secure if, for all polynomial-time attackers $\mathcal{A}$, the probability that $\mathcal{A}$ wins the above game is negligible as a function of the security parameter k.*

It is easy to see that a signcryption scheme that is both IND-CCA secure and INT-CCA secure maintains both the confidentiality and the integrity/authenticity of a message in the face of any attack. Therefore, we define:

Definition 6 (Insider security). *A signcryption scheme is said to be insider secure if it is IND-CCA secure and INT-CCA secure.*

[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 Hybrid Signcryption Schemes

If we attempt to apply the standard hybrid encryption paradigm to the problem of creating a signcryption scheme with insider security, then we run into the same problem as we encounter when we attempt to create a hybrid signature scheme. In other words, suppose that we assume that we can separate a hybrid signcryption scheme into a KEM and a DEM, where the generation-encryption algorithm for a message m runs as follows:

1. Execute the KEM on the input (sk_s, pk_r). It outputs a random symmetric key K and an encapsulation of that key C_1.
2. Encrypt the message m under the key K to produce a ciphertext C_2 using the DEM.
3. Output the signcryption (C_1, C_2).

In this case, an inside attacker who is able to obtain a valid signcryption (C_1, C_2) can forge a signcryption on any message m by recovering a symmetric key K from C_1 (using pk_s and sk_r), and encrypting the message m using the DEM and the symmetric key K.

For a hybrid signature scheme, the solution was to provide the KEM's encapsulation and decapsulation algorithm with the message as an extra input. However, for a hybrid signcryption scheme, we cannot provide the KEM's decapsulation oracle with the message as input, because the message has not yet been recovered at the time that we execute the decapsulation algorithm. On the other hand, it is necessary to make sure that the symmetric key used for decryption is related to the message being decrypted or we may still apply the simple forgery attack described above. We therefore define an insider secure KEM and DEM as follows.

Definition 7 (Signcryption KEM). *An (insider secure) 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 returns 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. *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 $(K, C) = Encap(sk_s, pk_r, m)$.*
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`*. We denote this algorithm as $Ver(pk_s, sk_r, m, C)$. 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), receiver key-pairs (pk_r, sk_r) and messages m then $K = Decap(pk_s, sk_r, C)$ for almost all pairs $(K, C) = Encap(sk_s, pk_r, m)$. We also require that the verification algorithm is sound, i.e. for almost all sender key-pairs (pk_s, sk_s), receiver key-pairs (pk_r, sk_r), messages m and encapsulations $(K, C) = Encap(sk_s, pk_r, m)$ then $Ver(pk_s, sk_r, m, C) =$ `valid`*.*

Definition 8 (Signcryption DEM). *A signcryption DEM consists of two polynomial-time 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 define a hybrid signcryption algorithm composed of a signcryption KEM and DEM in the following way.

Definition 9 (KEM-DEM hybrid signcryption scheme). *Suppose that $(Gen_c, Gen_s, Gen_r, Encap, Decap, Ver)$ 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 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 The Security Criteria for a Signcryption KEM

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

Confidentiality

A signcryption KEM must satisfy a similar condition to that satisfied by an encryption KEM [5, 9]. 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 $b = b'$. $\mathcal{A}$'s advantage in winning the IND-CCA2 game is defined to be:

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

Definition 10. *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, now, along with making sure that the keys that the signcryption KEM produces are suitably random, we must also now protect against the threat that a signcryption KEM leaks information about the message directly. To do this, we define a new game, the INP-CCA2 game[5] which states that an attacker cannot, given an encapsulation, distinguish between two messages that may have been used to produce it.

Formally, we define the INP-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.

[5] Here INP stands for "input".

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*.
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 decapsulation oracle on the input C^* or verification oracle on the inputs (m_0, C^*) or (m_1, C^*).

The attacker wins the game if $b = b'$. $\mathcal{A}$'s advantage in winning the INP-CCA2 game is defined to be:

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

Definition 11. *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.*

Integrity/Authenticity

It is clear that if an attacker, equipped with knowledge of pk_s, pk_r and sk_r, 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 (from which we somehow manage to derive a symmetric key) and a symmetric encryption scheme. 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. 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, pk_r, 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 $\mathit{Decap}(pk_s, sk_r, C^*) = K \neq \perp$, $\mathit{Ver}(pk_s, sk_r, m^*, C^*)$ outputs `valid`, and C^* was never the response from the encapsulation oracle queried on the message m. Note that we do not have to give the attacker explicit access to a decapsulation or verification oracle because they already know sk_r and can therefore compute these functions themselves.

Definition 12. *A signcryption KEM is INT-CCA2 secure if, for all polynomial-time attackers $\mathcal{A}$, the probability that $\mathcal{A}$ wins the INT-CCA2 game is negligible as a function of the security parameter k.*

Putting this all together we have:

Definition 13. *A signcryption KEM is said to be insider secure if it is IND-CCA2, INP-CCA2 and INT-CCA2 secure.*

6 The Security Criterion for a Signcryption DEM

As we have already noted, in a hybrid signcryption scheme with insider security, the KEM will be providing the integrity, origin authentication and non-repudiation services, so the DEM will only be required to provide a simple

confidentiality service. Indeed, the notion of security against passive attacks developed by Cramer and Shoup [5] is sufficient to provide security, with one slight exception.

Security in this model is phrased in terms of a game between a challenger and a two-stage attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. It runs as follows:

1. The challenger randomly generates a symmetric key K of the appropriate length for use with the symmetric encryption scheme.
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 $C^* = \text{ENC}_K(m_b)$.
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 $b = b'$. The attacker's advantage in winning this game is defined to be:

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

Definition 14. *A DEM is said to be secure against passive attacks if, for all polynomial time attackers $\mathcal{A}$, the advantage that $\mathcal{A}$ has in winning the above game is negligible as a function of the security parameter k.*

Since we require a signcryption scheme to have strong unforgeability, we actually require another property from the DEM. We require that the decryption algorithm is one-to-one[6], i.e. we require that, for any symmetric key K,

$$\text{DEC}_K(C_2) = \text{DEC}_K(C_2') \text{ if and only if } C_2 = C_2' \,. \tag{5}$$

This prevents an attacker from creating a forgery (C_1, C_2') from a signcryption (C_1, C_2) by finding another DEM encryption C_2' from the ciphertext C_2.

We can now state our main results.

Theorem 1 (Confidentiality). *Suppose a hybrid signcryption scheme is composed of a signcryption KEM and a signcryption DEM. If the signcryption KEM is insider secure and the DEM is secure against passive attacks and has a one-to-one decryption function, then the overall signcryption scheme is IND-CCA2 secure.*

Theorem 2 (Integrity/Authenticity). *Suppose a hybrid signcryption scheme is composed of a signcryption KEM and a signcryption DEM. If the signcryption KEM is INT-CCA2 secure and the DEM has a one-to-one decryption function, then the overall signcryption scheme is INT-CCA secure.*

The proofs of both of these theorems can be found in the full version of this paper [6].

[6] Technically, we actually only require a weaker condition: that the decryption algorithm is computationally one-to-one. I.e., that no polynomial-time attacker can find a second ciphertext C_2' given C_2.

7 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 [10]. We present the provably secure variant of Zheng's scheme proposed by Baek, Steinfeld and Zheng [3] 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 $p-1$ and g is an element of $\mathbb{Z}_p^*$ of order q.
- *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 $S = t/(R+s) \bmod q$.
  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 presented by Baek, Steinfeld and Zheng [3]. The scheme is secure in the random oracle model, under the Gap Diffie-Hellman assumption.

8 Conclusion

We have demonstrated that a signcryption scheme with insider security can be constructed using a hybrid approach, although this construction is significantly more complex than the "standard" hybrid construction. Furthermore, we have shown that the original signcryption scheme proposed by Zheng can be thought of as a hybrid signcryption scheme. Indeed, an examination of the literature shows that most signcryption schemes with insider security can be thought of as hybrid schemes in this form. This poses an interesting question: is it possible to construct a hybrid signcryption scheme that does not conform to the general model presented?

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 would also like to thank the anonymous reviewers for helpful comments. The author gratefully acknowledges the financial support of the EPSRC.

References

1. J. H. An. Authenticated encryption in the public-key setting: Security notions and analyses. Available from `http://eprint.iacr.org/2001/079`, 2001.
2. 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.
3. 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.
4. 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.
5. 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.
6. A. W. Dent. Hybrid cryptography. Available from `http://eprint.iacr.org/2004/210/`, 2004.
7. A. W. Dent. Hybrid signcryption schemes with outsider security (extended abstract). Available from `http://www.isg.rhul.ac.uk/~alex/`, 2004.
8. C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
9. 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.
10. 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.