The Cramer-Shoup Encryption Scheme is Plaintext Aware in the Standard Model (Version 3.0)

Alexander W. Dent

Information Security Group, Royal Holloway,
Egham, Surrey, TW20 0EX, U.K.
a.dent@rhul.ac.uk

Abstract. In this paper we examine the security criteria for a KEM and a DEM that are sufficient for the overall hybrid encryption scheme to be plaintext-aware in the standard model. We apply this theory to the Cramer-Shoup hybrid scheme acting on fixed length messages and deduce that the Cramer-Shoup scheme is plaintext-aware in the standard model. This answers a previously open conjecture of Bellare and Palacio on the existence of plaintext-aware encryption schemes.

1 Introduction

Plaintext awareness is a simple concept with a difficult explanation. An encryption scheme is plaintext aware if it is practically impossible for any entity to produce a ciphertext without knowing the associated message. This effectively renders a decryption oracle useless to an attacker, as any ciphertext submitted for decryption must either be invalid or the attacker must already know the decryption of that ciphertext and so does not gain any information by querying the oracle. Thus a scheme that is plaintext aware and semantically secure should be secure against adaptive attacks.

There are two problems with this simplistic approach. Firstly, if we wish to achieve the IND-CCA2 definition of security for an encryption scheme, then we have to be careful about how we define plaintext awareness, because, in this model, the attacker is always given one ciphertext for which he does not know the corresponding decryption (the challenge ciphertext). It is usually comparatively simple to achieve plaintext awareness when you do not have to consider the attacker as able to get hold of ciphertexts for which he does not know the corresponding decryption. We will follow the notation of Bellare and Palacio [4] and term this PA1 plaintext-awareness. A scheme that is IND-CPA and PA1 plaintext aware is only IND-CCA1 secure [4]. It is a lot harder to prove plaintext-awareness in full generality, when the attacker has access to an oracle that will return ciphertexts for which the attacker does not know the corresponding decryption, especially if the attacker has some measure of control over the probability distribution that the oracle uses to select the messages that it encrypts. This is termed PA2 plaintext awareness.

The importance of this issue was highlighted by the OAEP padding scheme [5]. This padding scheme was shown to be PA1 secure when combined with any one-way trapdoor permutation, however it was not recognised at the time that these scheme had to achieve PA2 plaintext awareness to guarantee full security [15]. The IND-CCA2 security of the OAEP padding scheme was finally demonstrated by Fujisaki *et al.* under a stronger security assumption [10]; however, the PA2 plaintext awareness of the OAEP padding scheme is still an open problem.

The second problem is that it is difficult to formally define plaintext awareness. The obvious way to define it is to say that for every attacker $\mathcal{A}$ that outputs a challenge ciphertext C, there exists a plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$ that outputs the decryption of C when given C as input. However, any encryption scheme that satisfies this definition of plaintext awareness in the standard model must necessarily fail to be IND-CPA secure. Hence, such a definition is not useful. For a satisfactory definition of plaintext awareness to be proposed, it is imperative that the plaintext extractor $\mathcal{A}^*$ be given some extra information about the actions that the attacker $\mathcal{A}$ took in order to compute the challenge ciphertext.

The original definition of plaintext awareness [2] was only given in the random oracle model and the plaintext extractor was given access to the oracle queries that the attacker made when constructing ciphertexts. This definition works well, but can only prove the security of a scheme in the random oracle model. Recently, Bellare and Palacio proposed a definition of plaintext awareness for the standard model [4]. In this model the plaintext extractor is given access to the random coins that the attacker used in constructing the challenge ciphertext; thus the plaintext extractor can examine every action that the attacker took in its execution. Unfortunately, Bellare and Palacio were unable to prove that any scheme met their strongest (PA2) definition of plaintext awareness, although they suggested that the Cramer-Shoup scheme [6] was a very likely candidate.

Organisation

This paper investigates plaintext-aware hybrid encryption, and, in particular, proves that the Cramer-Shoup encryption scheme is plaintext aware in the standard model. This proves a conjecture of Bellare and Palacio. The paper is organised as follows:

- Section 2 recalls security notions for secure asymmetric encryption schemes, hybrid (KEM-DEM) encryption schemes and plaintext-aware encryption schemes.
- Section 3 introduces two new techniques that will be later used to prove the security of the Cramer-Shoup encryption scheme: *encryption simulation* and *PA1+ plaintext awareness*. An encryption scheme that is simulatable is necessarily plaintext aware, and so we consider the concept to be of limited use. However, the concept of PA1+ plaintext awareness has a natural interpretation and may have further scope.
- Section 4 investigates sufficient conditions for a KEM and a DEM so that the resulting hybrid encryption scheme is plaintext aware. It should be noted that the results of this section are not contained in the conference proceedings version of this paper.

– Section 5 presents a proof that the Cramer-Shoup encryption scheme. The proof is presented under several computational assumptions, including the controversial Diffie-Hellman Knowledge (DHK) assumption. We also assume the existence of groups on which the DDH problem is hard and the existence of suitably secure hash functions.

2 Preliminaries

2.1 Asymmetric Encryption Schemes

We briefly recap the notion of an asymmetric cipher and of a KEM-DEM hybrid cipher [6]. We will assume that the reader is familiar with the general theory of hybrid ciphers and will concentrate on introducing notation that will be used in this paper. An asymmetric encryption scheme is a triple of algorithms:

1. A probabilistic polynomial-time *key generation algorithm*, $\mathcal{G}$, which takes as input a security parameter 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 that can be submitted to the encryption algorithm, and the *ciphertext space* $\mathcal{C}$, which is the set of all possible ciphertexts that can be submitted to the decryption algorithm (and may be larger than the range of the encryption algorithm).
2. A (possibly) probabilistic polynomial-time *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 $C = \mathcal{E}(pk, m)$.
3. A deterministic polynomial-time *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 $m = \mathcal{D}(sk, C)$.

The security of an asymmetric encryption scheme is assessed via the following game played between a two-stage attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a hypothetical challenger:

1. The challenger generates a valid public/private key pair (pk, sk) by running $\mathcal{G}(1^k)$.
2. The attacker runs $\mathcal{A}_1$ on the input pk. It terminates by outputting two equal-length messages m_0 and m_1, as well as some state information *state*. During its execution $\mathcal{A}_1$ may query an oracle $\mathcal{O}_1$.
3. The challenger picks a bit $b \in \{0, 1\}$ uniformly at random and computes the challenge ciphertext $C^* = \mathcal{E}(pk, m_b)$.
4. The attacker runs $\mathcal{A}_2$ on C^* and *state*. It terminates by outputting a guess b' for b. Again, during its executions, $\mathcal{A}_2$ may query an oracle $\mathcal{O}_2$ subject to the restriction that it may not query the oracle on the input C^*.

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

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

Throughout this paper NULL will be either the empty bit-string or the Turing machine that returns the empty bit-string for any input. We trust the reader will be able to differentiate between these two meanings by the context.

Definition 1. *Suppose, for all polynomial-time attackers $\mathcal{A}$, the advantage that $\mathcal{A}$ has in breaking the above game for an encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is negligible as a function of the security parameter k. The encryption scheme is said to be*

- *IND-CPA secure if $\mathcal{O}_1 = \mathcal{O}_2 =$ NULL.*
- *IND-CCA1 secure if $\mathcal{O}_1 = \mathcal{D}(sk, \cdot)$ and $\mathcal{O}_2 =$ NULL.*
- *IND-CCA2 secure if $\mathcal{O}_1 = \mathcal{O}_2 = \mathcal{D}(sk, \cdot)$.*

For more information on the basic security models for an asymmetric encryption scheme, the reader is referred to [2].

2.2 KEMs and DEMs

A hybrid cipher is an asymmetric cipher which uses a keyed symmetric algorithm, such as an encryption algorithm or a MAC, as a subroutine. Most hybrid ciphers can be presented as the combination of an asymmetric key encapsulation method (KEM) and a symmetric data encapsulation method (DEM). A KEM is a triple of algorithms consisting of:

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

The security of a KEM is phrased in terms of a game played between a hypothetical challenger and a two-stage attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. The attack goal for an attacker (against a KEM) is to distinguish the real key of an encapsulation from a randomly generated key. This is known as the IND 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 $\mathit{Gen}(1^k)$.
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 an oracle $\mathcal{O}_1$.
3. The challenger generates a valid encapsulation (C^*, K_0) by running $\mathit{Encap}(pk)$. 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 (C^*, K^*).

4. The attacker runs $\mathcal{A}_2$ on the input (C^*, K^*) and *state*. It terminates by outputting a guess b' for b. Again, during its executions, $\mathcal{A}_2$ may query an oracle $\mathcal{O}_2$ subject to the restriction that it may not query the oracle on the input C^*.

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

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

Definition 2. *Suppose, for all polynomial-time attackers $\mathcal{A}$, the advantage that $\mathcal{A}$ has in breaking the IND game for the KEM is negligible as a function of the security parameter k. The KEM is said to be*

- *IND-CPA secure if $\mathcal{O}_1 = \mathcal{O}_2 = \textsc{Null}$.*
- *IND-CCA1 secure if $\mathcal{O}_1 = Decap(sk, \cdot)$ and $\mathcal{O}_2 = \textsc{Null}$.*
- *IND-CCA2 secure if $\mathcal{O}_1 = \mathcal{O}_2 = Decap(sk, \cdot)$.*

A DEM is a pair of algorithms consisting of:

1. A deterministic, polynomial-time encryption algorithm, $\textsc{Enc}$, 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, polynomial-time decryption algorithm, $\textsc{Dec}$, 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 $\perp$.

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

1. The challenger randomly generates an appropriately sized symmetric key K. Note that the key length of K will depend on the security parameter k.
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 $(C^*, state)$. During its execution $\mathcal{A}_2$ may query on oracle $\mathcal{O}$. It may not query this oracle on the ciphertext C^*. This algorithm outputs a guess b' for b.

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

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

Definition 3. *Suppose for all polynomial-time attackers $\mathcal{A}$, the advantage that $\mathcal{A}$ has in breaking the IND game for the DEM is negligible as a function of the security parameter k. The DEM is said to be*

- *IND-PA (passively) secure if $\mathcal{O} = \textsc{Null}$.*

– *IND-CCA (actively) secure if* $\mathcal{O} = \textsc{Dec}_K(\cdot)$.

Theorem 1 (Cramer-Shoup). *A hybrid encryption scheme composed of an IND-CCA2 KEM and an IND-CCA DEM is IND-CCA2 as an encryption scheme.*

We will also require that our DEMs are in some way "plaintext aware" too, i.e. that it is impossible to find a valid DEM encryption except via an encryption oracle. This idea is capture by the INT-CCA+ model of security of Dent [8].

Definition 4. *INT-CCA+ security for a DEM* $(\textsc{Enc}, \textsc{Dec})$ *is defined using the following game played between a challenger and an attacker* $\mathcal{A}$*:*

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*. The DEM is said to be INT-CCA+ secure if, for every attacker, the probability that that attacker wins the INT-CCA+ game is negligible as a function of the security parameter.*

All of the standard DEM constructions, including the Encrypt-then-MAC scheme originally proposed as a DEM by Cramer and Shoup, are INT-CCA+ secure.

2.3 Plaintext-awareness

We use the notions and notations given by Bellare and Palacio [4]. The notion of plaintext awareness in the standard model states that an encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is plaintext aware if, for all ciphertext creators (attackers) $\mathcal{A}$, there exists a plaintext extractor $\mathcal{A}^*$ which takes as input the random coins of $\mathcal{A}$ and can answer the decryption queries of $\mathcal{A}$ in a manner that $\mathcal{A}$ cannot distinguish from a real decryption oracle. In order that $\mathcal{A}$ can be given access to ciphertexts for which it does not know the corresponding decryption, $\mathcal{A}$ will be allowed to query a plaintext creation oracle $\mathcal{P}$ with some query information *aux*. The plaintext creation oracle will pick a message at random (possibly from a distribution partially defined by *aux*) and returns the encryption of that message to the attacker[1]. Note that both $\mathcal{A}^*$ and $\mathcal{P}$ retain their state and random tape between invocations.

[1] Technically, the plaintext creator will only generate a random message, and it will be left to the challenger to compute the encryption of that message. However, since the ciphertext creator and the plaintext extractor receive exactly the same inputs regardless of whether the challenger or the plaintext creator encrypts the message, we do not distinguish between the two cases.

We will assume that all the algorithms described are polynomial-time, probabilistic, state-based Turing machines, and that the random coins of the Turing machine $\mathcal{A}$ are denoted $R[\mathcal{A}]$. Plaintext awareness is formally defined using two games. First we define the **REAL** game:

1. The challenger generates a random key pair $(pk, sk) = \mathcal{G}(1^k)$ and creates an empty list of ciphertexts CList.
2. The attacker executes $\mathcal{A}$ on pk.
   - If the attacker queries the encryption oracle with query information aux, then the challenger generates a random message $m = \mathcal{P}(aux)$ and computes its encryption $C = \mathcal{E}(pk, m)$. It adds C to CList and returns C to the attacker.
   - If the attacker queries the decryption oracle with a ciphertext C, then the decryption oracle returns $\mathcal{D}(sk, C)$. The attacker may not query the decryption oracle with any ciphertext appearing on CList.

   The attacker terminates by outputting a bitstring x.

The **FAKE** game is defined as:

1. The challenger generates a random key pair $(pk, sk) = \mathcal{G}(1^k)$ and creates an empty list of ciphertexts CList.
2. The attacker executes $\mathcal{A}$ on pk.
   - If the attacker queries the encryption oracle with query information aux, then the challenger generates a random message $m = \mathcal{P}(aux)$ and computes its encryption $C = \mathcal{E}(pk, m)$. It adds C to CList and returns C to the attacker.
   - If the attacker queries the decryption oracle with a ciphertext C, then the decryption oracle returns $\mathcal{A}^*(C, pk, R[\mathcal{A}], \text{CList})$. The attacker may not query the decryption oracle with any ciphertext appearing on CList.

   The attacker terminates by outputting a bitstring x.

Definition 5 (Plaintext awareness). *An asymmetric encryption scheme is said to be plaintext aware (PA2) if for all ciphertext creators $\mathcal{A}$, there exists a plaintext extractor $\mathcal{A}^*$ such that for all plaintext creators $\mathcal{P}$ and polynomial time algorithms Dist the advantage*

$$|Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{REAL}] - |Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{FAKE}]| \qquad (4)$$

*that Dist has in distinguishing whether $\mathcal{A}$ interacts with the **REAL** game or the **FAKE** game is negligible as a function of the security parameter.*

An asymmetric encryption scheme is said to be PA1 if for all ciphertext creators $\mathcal{A}$ that make no encryption oracle queries, there exists a plaintext extractor $\mathcal{A}^*$ such that for all polynomial time algorithms Dist the advantage

$$|Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{REAL}] - |Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{FAKE}]| \qquad (5)$$

is negligible as a function of the security parameter.

3 Simulatable Encryption Schemes

The aim of this paper is to show that the hybrid Cramer-Shoup scheme is plaintext-aware. In order to do this we take advantage of a very useful property that it possess: when instantiated with a suitable DEM, no attacker can distinguish valid ciphertexts from completely random bit strings. By this we mean that there exists a function f, which is in some sense invertible, that takes random bits as input and outputs bit strings that look like ciphertexts to an attacker. These bit strings are very unlikely to actually *be* valid ciphertexts (as we believe that the Cramer-Shoup scheme is plaintext aware) but no attacker can distinguish them from valid ciphertexts. We call this *encryption simulation*. For a simulatable encryption scheme, an attacker's ability to get hold of new ciphertexts in the PA2 model is equivalent to an ability to get hold of blocks of random data. A scheme that remains plaintext-aware even when the attacker can get hold new fixed-length random strings on demand is said to be PA1+ plaintext aware. This notion is stronger than PA1, but conceptually weaker than PA2 plaintext awareness.

3.1 Simulatable Encryption

We will wish to work with encryption schemes that are simulatable, by which we mean that there exists a polynomial-time Turing machine f which take a string of random bits as input and produces an output that cannot be distinguished from real ciphertexts. The difference between f and the real encryption function is that f must be in some sense invertible. We envisage f taking long strings of random bits as input and producing a shorter output, and so we insist on the existence of a polynomial-time Turing machine f^{-1} which acts as a perfect inverse for f when used on the right, i.e.

$$f(f^{-1}(C)) = C \text{ for all } C \in \mathcal{C}\,. \tag{6}$$

However, since f^{-1} cannot act as a perfect inverse for f when used on the left, we merely require that $f^{-1}(f(r))$ looks like a randomly generated bit string, i.e. it is computationally infeasible to tell the difference between a random string r of the appropriate length and $f^{-1}(f(r))$. Hence, f^{-1} must be a probabilistic polynomial-time Turing machine; while, for technical reasons, f must be a deterministic polynomial-time Turing machine.

Definition 6 (Simulatable Encryption Scheme). *An asymmetric encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is simulatable if there exist two polynomial-time Turing machines (f, f^{-1}) such that:*

- *f is a deterministic Turing machine that takes the public key pk and an element $r \in \{0,1\}^l$ as input, and outputs elements of $\mathcal{C}$. For simplicity's sake, we shall often represent f as a function from $\{0,1\}^l$ to $\mathcal{C}$ and suppress the public key input.*

- *f^{-1} is a probabilistic Turing machine that takes the public key pk and an element $C \in \mathcal{C}$ as input, and outputs elements of $\{0,1\}^l$. Again, we will often represent f^{-1} as a function from $\mathcal{C}$ to $\{0,1\}^l$ and suppress the public key input.*
- *$f(f^{-1}(C)) = C$ for all $C \in \mathcal{C}$.*
- *There exists no polynomial-time attacker $\mathcal{A}$ that has a non-negligible advantage in winning the following game:*
  1. *The challenger generates a key pair $(pk, sk) = \mathcal{G}(1^k)$ and randomly chooses a bit $b \in \{0,1\}$.*
  2. *The attacker executes $\mathcal{A}$ on the input pk. The attacker has access to an oracle $\mathcal{O}_f$ that takes no input, generates a random element $r \in \{0,1\}^l$, and returns r if $b = 0$ and $f^{-1}(f(r))$ if $b = 1$. The attacker terminates by outputting a guess b' for b.*

  The attacker wins if $b = b'$ and its advantage is defined in the usual way.
- *There exists no polynomial-time attacker $\mathcal{A}$ that has a non-negligible advantage in winning the following game:*
  1. *The challenger generates a key pair $(pk, sk) = \mathcal{G}(1^k)$, an empty list* CList*, and a bit b chosen randomly from $\{0,1\}$.*
  2. *The attacker executes $\mathcal{A}$ on the input pk. The attacker has access to two oracles:*
     * *An encryption oracle that takes a message $m \in \mathcal{M}$ as input and returns an encryption C. If $b = 0$, then the oracle returns $C = \mathcal{E}(pk, m)$. If $b = 1$, then the oracle returns $C = f(r)$, for some randomly chosen $r \in \{0,1\}^l$. In either case C is added to* CList.
     * *A decryption oracle that takes an encryption $C \in \mathcal{C}$ as input and returns $\mathcal{D}(sk, C)$. The attacker may not query the decryption oracle on any $C \in$* CList.

     The attacker terminates by outputting a guess b' for b.

  The attacker wins if $b = b'$ and its advantage is defined in the usual way.

At this stage, and for technical reasons that will become apparent in the next section, we will restrict ourselves to encryption schemes that have fixed-length ciphertext spaces, i.e. the ciphertext space $\mathcal{C} = \{0,1\}^n$ for some n. Normally, the simplest way of producing a cipher with fixed-length ciphertexts is to restrict the message space to fixed-length messages.

Theorem 2. *If $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is a simulatable encryption scheme then it is IND-CCA2 secure.*

Proof Let $\mathcal{A}$ be an IND-CCA2 attacker for the scheme, and let **Game 1** be the game in which $\mathcal{A}$ interacts with the IND-CCA2 game properly. Let **Game 2** be similar to Game 1 except that the challenge ciphertext is computed by applying f to a randomly generated string $r \in \{0,1\}^l$, rather than using the proper encryption algorithm. Let W_i be the event that $\mathcal{A}$ wins Game i.

Consider the following algorithm $\mathcal{B}$ against the simulatability of the encryption scheme:

1. The challenger generates a key pair $(pk, sk) = \mathcal{G}(1^k)$, an empty list CList, and a bit b chosen randomly from $\{0, 1\}$.
2. $\mathcal{B}$ executes $\mathcal{A}_1$ on the input pk. If $\mathcal{A}_1$ makes a decryption oracle query, then this is passed directly to $\mathcal{B}$'s decryption oracle and the result returned to $\mathcal{A}_1$. $\mathcal{A}_1$ terminates by outputting two equal-length messages m_0 and m_1, and some state information *state*.
3. $\mathcal{B}$ randomly chooses a bit $d \in \{0, 1\}$ and queries its encryption oracle with the message m_d. $\mathcal{B}$ receives back a ciphertext C^*.
4. $\mathcal{B}$ executes $\mathcal{A}_2$ on the input $(C^*, state)$. If $\mathcal{A}_2$ makes a decryption oracle query, then this is passed directly to $\mathcal{B}$'s decryption oracle and the result returned to $\mathcal{A}_2$. Note that $\mathcal{A}_2$ will never force $\mathcal{B}$ to make a decryption oracle query on $C^* \in$ CList due to the nature of the IND-CCA2 game. $\mathcal{A}_2$ terminates by outputting a guess d' for d.
5. If $d = d'$, then $\mathcal{B}$ outputs 1. Otherwise $\mathcal{B}$ outputs 0.

If $b = 0$ then $\mathcal{B}$ perfectly simulates Game 1 for $\mathcal{A}$. If $b = 1$ then $\mathcal{B}$ perfectly simulates Game 2 for $\mathcal{A}$. In both cases $\mathcal{B}$ outputs 1 if and only if $\mathcal{A}$ wins. It is well known that we may express $\mathcal{B}$'s advantage as:

$$\frac{1}{2} |Pr[\mathcal{B} \text{ outputs } 1 | b = 0] - Pr[\mathcal{B} \text{ outputs } 1 | b = 1]| \,. \tag{7}$$

However,

$$|Pr[\mathcal{B} \text{ outputs } 1 | b = 0] - Pr[\mathcal{B} \text{ outputs } 1 | b = 1]| = |Pr[W_0] - Pr[W_1]| \,. \tag{8}$$

Hence, $|Pr[W_1] - Pr[W_2]|$ is negligible, as the encryption algorithm is simulatable. In Game 2, though, the challenge ciphertext is completely independent of the messages supplied by the attacker. Therefore, $Pr[W_2] = 1/2$ and $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is IND-CCA2 secure. □

Therefore, in some sense, the notion of encryption simulation is less useful than one might hope. It should be easier to prove that a scheme is IND-CCA2 secure, than to show that it is simulatable; and if we can show that a scheme is simulatable, then there is no need to consider whether it is plaintext aware, as we have already shown that it is IND-CCA2. However, since the goal of this paper is merely to show that PA2 schemes exist, we will continue to consider the notion of simulatability.

3.2 PA1+ Plaintext Awareness

For a simulatable encryption algorithm, a ciphertext creator's ability to get hold of new, randomly generated ciphertexts C (that are the encryption of messages drawn from some distribution) is equivalent to being able to get hold of randomly generated strings $r = f^{-1}(C) \in \{0, 1\}^l$. We define the PA1+ model as the extension of the PA1 model in which a ciphertext creator has access to an oracle which provides it with randomly generated bit strings of length l, and show that, for a simulatable encryption algorithm, this is enough to imply that the scheme is PA2 plaintext-aware.

We define the PA1+ model using the **REAL** and **FAKE** games as before. For an attacker $\mathcal{A}$ and a hypothetical challenger, the **REAL** game works as follows:

1. The challenger generates a random key pair $(pk, sk) = \mathcal{G}(1^k)$.
2. The attacker executes $\mathcal{A}$ on pk. The attacker has access to a decryption oracle and to a randomness oracle.
   – If the attacker queries the randomness oracle, then the challenger generates a random strong $r \in \{0,1\}^l$, and returns r to the attacker.
   – If the attacker queries the decryption oracle with a ciphertext C, then the decryption oracle returns $\mathcal{D}(sk, C)$.

   The attacker terminates by outputting a bitstring x.

The **FAKE** game is defined in the obvious way:

1. The challenger generates a random key pair $(pk, sk) = \mathcal{G}(1^k)$ and creates an (empty) list of random blocks RList.
2. The attacker executes $\mathcal{A}$ on pk. The attacker has access to a decryption oracle and to a randomness oracle.
   – If the attacker queries the randomness oracle, then the challenger generates a random strong $r \in \{0,1\}^l$, adds r to RList and returns r to the attacker.
   – If the attacker queries the decryption oracle with a ciphertext C, then the decryption oracle returns $\mathcal{A}^*(C, pk, R[\mathcal{A}], \mathrm{RList})$.

   The attacker terminates by outputting a bitstring x.

Definition 7 (PA1+ Plaintext Awareness). *An asymmetric encryption scheme is said to be PA1+ plaintext aware if for all polynomial-time ciphertext creators $\mathcal{A}$, there exists a polynomial-time plaintext extractor $\mathcal{A}^*$ such that for all polynomial-time distinguishing algorithms Dist the advantage*

$$|Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{\textit{REAL}}] - |Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{\textit{FAKE}}]| \tag{9}$$

*that Dist has in distinguishing whether $\mathcal{A}$ interacts with the **REAL** game or the **FAKE** game is negligible as a function of the security parameter.*

Intuitively, the difference between PA1 and PA1+ is in the ability for the ciphertext creator to act in a manner that is unpredictable by the plaintext extractor after the plaintext extractor has returned a message. For a scheme that is PA1, the plaintext extractor, when attempting to provide some sort of decryption of a ciphertext, knows exactly what the ciphertext creator is going to do with the ciphertext (as it has access to the ciphertext creator's random tape). Hence, the plaintext creator can tailor its response to make sure that that particular execution of the ciphertext creator cannot differentiate between the plaintext extractor's response and the response of a real decryption oracle. However, a PA1+ ciphertext creator has the ability to acquire random bits that could affect its execution *after* it has received the plaintext extractor's response, and so the plaintext extractor cannot tailor its response in the same way.

3.3 The relationship between PA1, PA1+ and PA2

It is trivial to see that any scheme that is PA1+ plaintext aware is necessarily PA1 plaintext aware. However, it is not so easy to see the other relationships between the notions of plaintext awareness. In particular, whilst it seems unlikely that a scheme that is PA1 secure is necessarily PA1+ secure, this fact seems difficult to prove. In order to exhibit a scheme that is PA1 secure but not PA1+ secure, we have to find a scheme for which we can successfully deceive a ciphertext creator *if and only if* we know what the ciphertext creator's future actions are going to be. The construction of such a scheme is non-trivial given current knowledge about non-black-box attackers.

Conjecture 3 *If an asymmetric encryption scheme* $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ *is PA2, then it is PA1+.*

Evidence in support of conjecture We now provide evidence that this conjecture is true, although we will stop short of a full, formal proof. Let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a PA2 plaintext-aware encryption scheme and let $\mathcal{A}$ be a PA1+ ciphertext creator. The idea is to show that we can build a PA2 ciphertext creator $\bar{\mathcal{A}}$ that runs $\mathcal{A}$ as a subroutine and uses its access to an encryption oracle to provide new random bits to $\mathcal{A}$. We will then use the plaintext extractor $\bar{\mathcal{A}}^*$ for $\bar{\mathcal{A}}$ to produce a plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$. In our attempt to do this we note that

- We may assume (without loss of generality) that $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ has a message space $\mathcal{M} = \{0, 1\}$.
- We may choose the plaintext creator that $\bar{\mathcal{A}}$ is going to use when querying the encryption oracle. We set this to be the plaintext creator that choose the message 0 or 1 uniformly at random.
- Since $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is suppose to be a polynomial-time encryption scheme, we may assume (without loss of generality) that there exists a polynomial p such that the ciphertext space $\mathcal{C} \subseteq \{0, 1\}^{p(k)}$.
- We may assume (without loss of generality) that $\mathcal{A}$ interacts with a randomness oracle that returns a single random bit. We may construct larger blocks of random bits by repeatedly calling such an oracle.

Our strategy will be for $\bar{\mathcal{A}}$ to run $\mathcal{A}$ and answer all $\mathcal{A}$'s decryption queries using its own decryption oracle. The subtlety comes in how we deal with $\mathcal{A}$'s requests for new bits of randomness. Prior to invoking $\mathcal{A}$, $\bar{\mathcal{A}}$ splits it's random tape into two halves, R_1 and R_2. This can easily be achieved by viewing odd numbered squares as being part of R_1 and even numbered squares as being part of R_2. It then uses R_2 to generate a bitstring Δ of length $p(k)$ that consists of a single 1-bit placed in a random position. It sets R_1 to be the random tape of $\mathcal{A}$. If $\mathcal{A}$ requests a new bit of randomness, then $\bar{\mathcal{A}}$ queries its encryption oracle, receives a ciphertext C, and computes the inner product (modulo 2) of C with Δ.

We claim that there is a non-negligible probability that $C \cdot \Delta \bmod 2$ is 0 and a non-negligible probability that $C \cdot \Delta \bmod 2$ is 1. Suppose this was not the case, then, for almost all public keys pk, and every bit position $1 \leq i \leq p(k)$, there must exist a bit $b(i)$ such that the probability that the i-th bit of $\mathcal{E}(b, pk)$ is equal to $b(i)$ is (negligibly close to) 1, where b is a bit chosen at random. Hence, for large enough values of the security parameter, the encryption algorithm will output the bitstring $b(1)b(2)b(3)\ldots b(p(k))$ with overwhelming probability. However, this is a contradiction, as it is impossible for $b(1)b(2)b(3)\ldots b(p(k))$ to be an encryption of both 0 and 1. Hence, with non-negligible probability, $C \cdot \Delta \bmod 2$ provides a (biased) random source of bits.

$\bar{\mathcal{A}}$ can use standard probability amplification techniques to compute an (almost) unbiased bit of randomness from this biased random source. Hence, $\bar{\mathcal{A}}$ can, with non-negligible probability, correctly respond to randomness queries made by $\mathcal{A}$.

Since $\bar{\mathcal{A}}$ is a PA2 ciphertext creator, there exists a plaintext extractor $\bar{\mathcal{A}}^*$ for $\bar{\mathcal{A}}$. This algorithm takes as input the public key pk, the ciphertexts CList that $\bar{\mathcal{A}}$ has received from its encryption oracle, and the two random tapes R_1 and R_2 that $\bar{\mathcal{A}}$ uses. Note that $\bar{\mathcal{A}}^*$ is actually responding to the decryption oracle queries of $\mathcal{A}$ as all $\bar{\mathcal{A}}$ does is forward these queries.

We now show that we may construct a plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$ using $\bar{\mathcal{A}}^*$. $\mathcal{A}^*$ takes as input the public key pk, the list of randomness RList that $\mathcal{A}$ has received, and the random tape $R[\mathcal{A}]$. $\mathcal{A}^*$ will run $\bar{\mathcal{A}}^*$ as a subroutine, hence we need to simulate $\bar{\mathcal{A}}^*$'s inputs. We set R_1 to be the random tape of $R[\mathcal{A}]$. $\mathcal{A}^*$ randomly generates the tape R_2 and uses this to compute the bitstring Δ. Now, for every bit b in RList we construct a sequence of ciphertexts C to append to CList such that if the encryption oracle had returned these ciphertexts, then $\bar{\mathcal{A}}$ would have returned the bit b to $\mathcal{A}$. $\mathcal{A}^*$ does this by randomly generating ciphertexts in the same way as the encryption oracle and using Δ to extract a random bit from them. This bit will be equal to b with probability (about) one half; hence, we can (with non-negligible probability) construct a valid list of ciphertexts CList.

Given the inputs pk, CList, R_1 and R_2, $\bar{\mathcal{A}}^*$ will be able to correctly respond to the decryption oracle queries made by $\mathcal{A}$. Hence, there exists a PA1+ plaintext extractor and the scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is PA1+ plaintext aware. □

Hence, it is very likely that any scheme that is PA2 is necessarily PA1+. It seems very unlikely that the reverse implication holds, i.e. that any scheme that is PA1+ is necessarily PA2. However, the next theorem (which will be crucial in proving the Cramer-Shoup scheme is plaintext aware) gives a partial result.

Theorem 4. *Let $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ be a simulatable encryption algorithm. If $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is PA1+ then it is PA2.*

Proof This proof works in several stages. We wish to show that for any ciphertext creator for the hybrid encryption scheme $\mathcal{A}$, there exists a plaintext extractor $\mathcal{A}^*$. First we show that any PA2 ciphertext creator $\mathcal{A}$ for the encryption scheme can be used to create a PA1+ ciphertext creator $\bar{\mathcal{A}}$. Since the encryption scheme is PA1+ plaintext aware, there exist a plaintext extractor $\bar{\mathcal{A}}^*$ for $\bar{\mathcal{A}}$. We then show that we can use the plaintext extractor $\bar{\mathcal{A}}^*$ for $\bar{\mathcal{A}}$ to build a plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$. We will use this technique liberally throughout this paper.

Let $\mathcal{A}$ be any PA2 ciphertext creator and let $\bar{\mathcal{A}}$ be the PA1+ ciphertext creator that runs as follows.

1. Execute $\mathcal{A}$.
   - If $\mathcal{A}$ makes a decryption oracle query, then $\bar{\mathcal{A}}$ passes this query directly on to its own decryption oracle.
   - If $\mathcal{A}$ makes an encryption oracle query (with query information *aux*), then $\bar{\mathcal{A}}$ queries its randomness oracle, receives back an l-bit block of randomness r, and returns $f(r)$ to $\mathcal{A}$.
2. $\mathcal{A}$ terminates by outputting a bitstring x. Output x.

Let $W_{0,Dist}$ be the event that $Dist(x) = 1$ when $\mathcal{A}$ interacts with the PA2 model and a real decryption oracle. Let $W_{1,Dist}$ be the event that $Dist(x) = 1$ when $\bar{\mathcal{A}}$ interacts with the PA1+ model and a real decryption oracle. It is clear that any non-negligible difference between $Pr[W_{0,Dist}]$ and $Pr[W_{1,Dist}]$ can be used to create an algorithm that can distinguish between ciphertexts and simulated ciphertexts, contravening the final point of Definition 6. Thus,

$$|Pr[W_{0,Dist}] - Pr[W_{1,Dist}]|$$

is negligible as a function of the security parameter.

Since $\bar{\mathcal{A}}$ is PA1+ ciphertext creator, there exists a plaintext extractor $\bar{\mathcal{A}}^*$ for $\bar{\mathcal{A}}$. Let $W_{2,Dist}$ be the event that $Dist(x) = 1$ when $\bar{\mathcal{A}}$ interacts with the PA1+ model and $\bar{\mathcal{A}}^*$ is used to simulate the decryption oracle. Since $\bar{\mathcal{A}}^*$ is a successful plaintext extractor for $\bar{\mathcal{A}}$, we have that

$$|Pr[W_{1,Dist}] - Pr[W_{2,Dist}]|$$

is negligible as a function of the security parameter.

We now alter slightly the way that the randomness oracle works. Instead of randomly generated a block of randomness r and returning this to $\bar{\mathcal{A}}$, consider an oracle that randomly generates a block of randomness $r \in \{0,1\}^l$ and returns $f^{-1}(f(r))$ to the ciphertext creator. Let $W_{3,Dist}$ be the event that $Dist(x) = 1$ when the randomness oracle behaves in this way. Clearly, any significant difference between $Pr[W_{2,Dist}]$ and $Pr[W_{3,Dist}]$ can be used to create an algorithm that can distinguish between random blocks r and $f^{-1}(f(r))$, thus contravening the properties of f given in Definition 6. Hence,

$$|Pr[W_{2,Dist}] - Pr[W_{3,Dist}]|$$

is negligible as a function of the security parameter.

If we examine the architecture now, we notice that RLIST contains elements of the form $f^{-1}(f(r))$, and $\mathcal{A}$ (being run as a subroutine of $\bar{\mathcal{A}}$) is given elements of the form $f(f^{-1}(f(r))) = f(r)$. Consider now a situation where

- the randomness oracle returns $f(r)$ instead of $f^{-1}(f(r))$,
- to the ciphertext creator $\mathcal{A}$ (instead of $\bar{\mathcal{A}}$)

– and decryption queries are answered using a plaintext extractor $\mathcal{A}^*$. $\mathcal{A}^*$ works by executing $\bar{\mathcal{A}}^*$ on the input $(pk, C, R[\mathcal{A}], \text{RList})$, where C is the ciphertext to be decrypted and RList is the list of l-bit random blocks given by taking the responses C' returned the randomness oracle and computing $f^{-1}(C')$.

Let $W_{4,Dist}$ be the event that $Dist(x) = 1$ in this model. Clearly, the functionality of this model is identical to the previous model. Hence,

$$Pr[W_{3,Dist}] = Pr[W_{4,Dist}] .$$

We may now consider the model in which the randomness oracle reverts to being an encryption oracle. I.e. instead of returning $f(r)$ for some randomly chosen l-bit block r, it returns the encryption $\mathcal{E}(m, pk)$ for message $m = \mathcal{P}(aux)$. Let $W_{5,Dist}$ be the event that $Dist(x) = 1$ in this model. As before, if there is any significant difference between $Pr[W_{4,Dist}]$ and $Pr[W_{5,Dist}]$, then we may build an algorithm that distinguishes between ciphertexts and simulated ciphertexts, contravening Definition 6. Therefore,

$$|Pr[W_{4,Dist}] - Pr[W_{5,Dist}]|$$

is negligible. However, this means that

$$|Pr[W_{0,Dist}] - Pr[W_{5,Dist}]|$$

is negligible as a function of the security parameter, and so that $\mathcal{A}$ has a successful plaintext extractor $\mathcal{A}^*$. Therefore, $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is PA2 plaintext aware. □

3.4 Simulatable KEMs and DEMs

Since we wish to prove the security of the hybrid Cramer-Shoup scheme, we need to show that the hybrid scheme is simulatable if and only if the KEM and DEM of which it is formed is (in some sense) simulatable.

Definition 8 (Simulatable KEM). *A KEM $(Gen, Encap, Decap)$ is simulatable if there exist two polynomial-time Turing machines (f, f^{-1}) such that:*

- *f is a deterministic Turing machine that takes the public key pk and an element $r \in \{0,1\}^l$ as input, and outputs elements of $\mathcal{C}$. For simplicity's sake, we shall often represent f as a function from $\{0,1\}^l$ to $\mathcal{C}$ and suppress the public key input.*
- *f^{-1} is a probabilistic Turing machine that takes the public key pk and an element $C \in \mathcal{C}$ as input, and outputs elements of $\{0,1\}^l$. Again, we will often represent f^{-1} as a function from $\mathcal{C}$ to $\{0,1\}^l$ and suppress the public key input.*
- *$f(f^{-1}(C)) = C$ for all $C \in \mathcal{C}$.*
- *There exists no polynomial-time attacker $\mathcal{A}$ that has a non-negligible advantage in winning the following game:*

1. *The challenger generates a key pair* $(pk, sk) = Gen(1^k)$ *and randomly chooses a bit* $b \in \{0, 1\}$.
2. *The attacker executes* $\mathcal{A}$ *on the input* pk. *The attacker has access to an oracle* $\mathcal{O}_f$ *that takes no input, generates a random element* $r \in \{0,1\}^l$, *and returns* r *if* $b = 0$ *and* $f^{-1}(f(r))$ *if* $b = 1$. *The attacker terminates by outputting a guess* b' *for* b.

The attacker wins if $b = b'$ *and its advantage is defined in the usual way.*

– *There exists no polynomial-time attacker* $\mathcal{A}$ *that has a non-negligible advantage in winning the following game:*
  1. *The challenger generates a key pair* $(pk, sk) = \mathcal{G}(1^k)$, *an empty list* CList, *and a bit* b *chosen randomly from* $\{0, 1\}$.
  2. *The attacker executes* $\mathcal{A}$ *on the input* pk. *The attacker has access to two oracles:*
     * *An encapsulation oracle that takes no input and returns a pair* (C, K). *If* $b = 0$, *then the oracle returns* $(C, K) = Encap(pk)$. *If* $b = 1$, *then the oracle returns* $C = f(r)$, *for some randomly chosen* $r \in \{0,1\}^l$, *and a randomly chosen symmetric key* K *of the appropriate size. In either case* C *is added to* CList.
     * *A decapsulation oracle that takes an encapsulation* $C \in \mathcal{C}$ *as input and returns* $Decap(sk, C)$. *The attacker may not query the decapsulation oracle on any* $C \in$ CList.

     The attacker terminates by outputting a guess b' *for* b.

  The attacker wins the game if $b = b'$ *and its advantage is defined in the usual way.*

Note that, just as in Theorem 2 for encryption schemes, a simulatable KEM is necessarily IND-CCA2 secure. We know turn our attention to the DEMs (with fixed length ciphertext spaces):

Definition 9 (Simulatable DEM). *A DEM* (Enc, Dec) *is simulatable if there exist two polynomial-time Turing machines* (f, f^{-1}) *such that:*

– f *is a deterministic Turing machine that takes an element* $r \in \{0,1\}^l$ *as input, and outputs elements of* $\mathcal{C}$. *For simplicity's sake, we shall often represent* f *as a function from* $\{0,1\}^l$ *to* $\mathcal{C}$.
– f^{-1} *is a probabilistic Turing machine that takes an element* $C \in \mathcal{C}$ *as input, and outputs elements of* $\{0,1\}^l$. *Again, we will often represent* f^{-1} *as a function from* $\mathcal{C}$ *to* $\{0,1\}^l$.
– $f(f^{-1}(C)) = C$ *for all* $C \in \mathcal{C}$.
– *There exists no polynomial-time attacker* $\mathcal{A}$ *that has a non-negligible advantage in winning the following game:*
  1. *The challenger randomly chooses a bit* $b \in \{0, 1\}$.
  2. *The attacker executes* $\mathcal{A}$ *on the input* 1^k. *The attacker has access to an oracle* $\mathcal{O}_f$ *that takes no input, generates a random element* $r \in \{0,1\}^l$, *and returns* r *if* $b = 0$ *and* $f^{-1}(f(r))$ *if* $b = 1$. *The attacker terminates by outputting a guess* b' *for* b.

  The attacker wins if $b = b'$ *and its advantage is defined in the usual way.*

– *There exists no polynomial-time attacker $\mathcal{A}$ that has a non-negligible advantage in winning the following game:*
  1. *The challenger generates a symmetric key K of the appropriate length, an empty list* CList*, and a bit b chosen randomly from $\{0, 1\}$.*
  2. *The attacker executes $\mathcal{A}$ on the input 1^k. The attacker has access to two oracles:*
     * *An encryption oracle that takes a message m input and returns a ciphertext C. If $b = 0$, then the oracle returns $C = \text{Enc}_K(m)$. If $b = 1$, then the oracle returns $C = f(r)$, for some randomly chosen $r \in \{0, 1\}^l$. In either case C is added to* CList.
     * *A decryption oracle that takes a ciphertext $C \in \mathcal{C}$ as input and returns $\text{Dec}_K(C)$. The attacker may not query the decryption oracle on any $C \in$* CList.

     The attacker terminates by outputting a guess b' for b.

  The attacker wins if $b = b'$ and its advantage is defined in the usual way.

Let (Kf, Kf^{-1}) be the Turing machines that are used to simulate a KEM, and (Df, Df^{-1}) be the Turing machines that are used to simulate a DEM. We may define a pair of Turing machines (f, f^{-1}) to simulate the overall hybrid encryption scheme as follows. Let f be the Turing machine take two inputs r_1 and r_2, where r_1 is the correct length to be used with Kf and r_2 is the correct length to be used with Df, and outputs $(Kf(r_1), Df(r_2))$. Similarly, let f^{-1} act on ciphertexts $(C_1, C_2) \in \mathcal{C}$ by outputting $(Kf^{-1}(C_1), Df^{-1}(C_2))$.

Lemma 1. *A hybrid encryption scheme composed of a simulatable KEM and a simulatable DEM is simulatable as an encryption scheme.*

The proof of this lemma follows easily from the definitions.

4 Plaintext-Aware KEMs

In this section we will develop criteria for a KEM and a DEM that are sufficient to guarantee that a KEM-DEM hybrid encryption scheme is plaintext aware. We will not claim that these are necessary conditions: it seems perfectly plausible that there exist plaintext-aware KEM-DEM encryption schemes that do not have plaintext-aware KEMs or DEMs. Indeed, we rather hope that this is the case, as our definitions will require that a hybrid encryption scheme is IND-CCA2 *before* we attempt to prove that it is fully plaintext aware.

4.1 Partial (PA1) Plaintext-Awareness

We separate the (relatively simple) PA1 case from the more complex PA2 case. We would expect the KEM to be PA1 if no ciphertext creator can produce an encapsulation that its associated plaintext extractor cannot decapsulate. Consider a KEM $(\mathit{Gen}, \mathit{Encap}, \mathit{Decap})$ with a ciphertext creator $\mathcal{A}$ and associated plaintext extractor $\mathcal{A}^*$. Formally, we define this notion using **REAL** and **FAKE** games again. We define the **REAL** game as follows:

1. The challenger generates a random key pair $(pk, sk) = Gen(1^k)$.
2. The attacker executes $\mathcal{A}$ on pk. If the attacker queries the decapsulation oracle with a ciphertext C, then the decapsulation oracle returns $Decap(sk, C)$. The attacker terminates by outputting a bitstring x.

The **FAKE** game is defined as follows:

1. The challenger generates a random key pair $(pk, sk) = Gen(1^k)$.
2. The attacker executes $\mathcal{A}$ on pk. If the attacker queries the decapsulation oracle with a ciphertext C, then the decapsulation oracle returns $\mathcal{A}^*(C, pk, R[\mathcal{A}])$. The attacker terminates by outputting a bitstring x.

Definition 10. *A KEM is said to be PA1 if, for all ciphertext creators $\mathcal{A}$, there exists a plaintext extractor $\mathcal{A}^*$ such that for all polynomial time distinguishers Dist the advantage*

$$|Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{\textit{REAL}}] - |Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{\textit{FAKE}}]| \tag{10}$$

*that Dist has in distinguishing whether $\mathcal{A}$ interacts with the **REAL** game or the **FAKE** game is negligible as a function of the security parameter.*

Theorem 5. *A hybrid encryption scheme composed of a PA1 KEM and an arbitrary DEM is PA1.*

Proof We show that any ciphertext creator $\mathcal{A}$ for the encryption scheme can be used to create a ciphertext creator $\bar{\mathcal{A}}$ for the KEM. Since the KEM is plaintext aware, there exists a plaintext extractor $\bar{\mathcal{A}}^*$ for $\bar{\mathcal{A}}$. We then use $\bar{\mathcal{A}}^*$ to construct a plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$.

Let $\mathcal{A}$ be a ciphertext creator for the hybrid encryption scheme. We define the ciphertext creator $\bar{\mathcal{A}}$ for the KEM as the algorithm that executes $\mathcal{A}$. If $\mathcal{A}$ queries the decryption oracle with a ciphertext (C_1, C_2), then $\bar{\mathcal{A}}$ queries the decapsulation oracle with encapsulation C_1. If the oracle returns $\perp$ then $\bar{\mathcal{A}}$ returns $\perp$ to $\mathcal{A}$. Otherwise the oracle returns a key K and $\bar{\mathcal{A}}$ returns $\text{Dec}_K(C_2)$ to $\mathcal{A}$.

Since $\bar{\mathcal{A}}$ is a valid ciphertext creator for the KEM, there exists a plaintext extractor $\bar{\mathcal{A}}^*$. We define a plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$ as follows. On the submission of a ciphertext (C_1, C_2), $\mathcal{A}^*$ executes $\bar{\mathcal{A}}^*$ on C_1. If $\bar{\mathcal{A}}^*$ returns $\perp$, then $\mathcal{A}^*$ returns $\perp$ to $\mathcal{A}$. Otherwise $\bar{\mathcal{A}}^*$ returns a key K, and $\bar{\mathcal{A}}^*$ returns $\text{Dec}_K(C_2)$. It is easy to see that the system in which $\mathcal{A}$ interacts with its decryption oracle (in the **REAL** or **FAKE** game) is the same as $\bar{\mathcal{A}}$ interacting with its decryption oracle in the same game. Hence, the outputs of $\mathcal{A}$ must be indistinguishable regardless of the game which $\mathcal{A}$ is playing. □

Therefore, using the result of Bellare and Palacio [4], we have that:

Corollary 1. *A hybrid encryption scheme composed of an IND-CPA and PA1 KEM, and an IND-PA DEM, is IND-CCA1 secure.*

Theorem 5 also provides necessary conditions.

Theorem 6. *If the family of hybrid encryption schemes created by composing a KEM $(Gen, Encap, Decap)$ with any arbitrary DEM (that takes keys of the length produced by the KEM) are all PA1, then the KEM is PA1.*

Proof Consider the DEM $(\textsc{Enc}, \textsc{Dec})$ given by

$$\textsc{Enc}_K(m) = m \oplus K \qquad \textsc{Dec}_K(C) = C \oplus K\,. \tag{11}$$

The hybrid encryption scheme formed by composing the KEM $(Gen, Encap, Decap)$ with this DEM is PA1. Let $\mathcal{A}$ be any ciphertext creator for the KEM, and define $\bar{\mathcal{A}}$ to be the ciphertext creator for the overall hybrid encryption scheme that runs as follows.

1. Execute $\mathcal{A}$. If $\mathcal{A}$ requests the decapsulation of the encapsulation C_1, then $\bar{\mathcal{A}}$ submits a the ciphertext $(C_1, 0^l)$ to the decryption oracle, where l is the (pre-defined) length of the symmetric keys produced by the KEM. The oracle responds with a message m. $\bar{\mathcal{A}}$ returns m to $\mathcal{A}$.
2. $\mathcal{A}$ terminates by outputting a bit-string x. $\bar{\mathcal{A}}$ terminates and outputs x.

Since $m = 0^l \oplus K = K$, it is clear that if there exists a successful plaintext extractor $\bar{\mathcal{A}}^*$ for $\bar{\mathcal{A}}$, then there exists a successful plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$ which runs as follows.

1. Execute $\bar{\mathcal{A}}^*$ on the input $(C, 0^l)$ and receive a value m from this subroutine.
2. Output m.

$\square$

4.2 Intermediate (PA1+) Plaintext-Awareness

In order to show that the Cramer-Shoup scheme is PA2 plaintext aware, we will show that it is PA1+ plaintext aware and that it is simulatable. In this section, we will demonstrate analogous results to the previous section, and show that a KEM/DEM scheme composed of a PA1+ KEM and an arbitrary DEM is PA1+.

We start by defining what we mean by a PA1+ KEM. The PA1+ model is the obvious extension of the PA1 model given by allowing the ciphertext creator access to a randomness oracle that returns fixed-length random strings. Formally, we define the **REAL** game as:

1. The challenger generates a random key pair $(pk, sk) = Gen(1^k)$.
2. The attacker executes $\mathcal{A}$ on pk.
   - If the attacker queries the randomness oracle, then the oracle generates a fixed-length random string $r \in \{0,1\}^l$ uniformly at random and returns r to the attacker.
   - If the attacker queries the decapsulation oracle with a ciphertext C, then the decapsulation oracle returns $Decap(sk, C)$.

   The attacker terminates by outputting a bitstring x.

The **FAKE** game is defined as follows:

1. The challenger generates a random key pair $(pk, sk) = Gen(1^k)$.
2. The attacker executes $\mathcal{A}$ on pk.
   - If the attacker queries the randomness oracle, then the oracle generates a fixed-length random string $r \in \{0,1\}^l$ uniformly at random, adds r to RList and returns r to the attacker.
   - If the attacker queries the decapsulation oracle with a ciphertext C, then the decapsulation oracle returns $\mathcal{A}^*(C, pk, R[\mathcal{A}], \text{RList})$.

   The attacker terminates by outputting a bitstring x.

Definition 11. *A KEM is said to be PA1+ if, for all ciphertext creators $\mathcal{A}$, there exists a plaintext extractor $\mathcal{A}^*$ such that for all polynomial time distinguishers Dist the advantage*

$$|Pr[Dist(x) = 1|\mathcal{A} \textit{ plays } \boldsymbol{REAL}] - |Pr[Dist(x) = 1|\mathcal{A} \textit{ plays } \boldsymbol{FAKE}]| \qquad (12)$$

*that Dist has in distinguishing whether $\mathcal{A}$ interacts with the **REAL** game or the **FAKE** game is negligible as a function of the security parameter.*

Theorem 7. *A hybrid encryption scheme composed of a PA1+ KEM and an arbitrary DEM is PA1+.*

The proof of this theorem can easily be adapted from the proof of Theorem 5.

4.3 Full (PA2) Plaintext-Awareness

The situation becomes more complex when we try to consider the use of an encapsulation oracle. Stam [16] considers an encapsulation oracle that computes $(C, K) = Encap(pk)$ and returns C to the ciphertext creator $\mathcal{A}$. Using the definition of PA2 given by augmenting the PA1 game with this encapsulation oracle, he was able to show that:

Theorem 8 (Stam). *A KEM that is IND-CPA and PA2 (in the random oracle model) is IND-CCA2 secure.*

This does not seem to be sufficient for our purposes. Instead we introduce a new concept, which we term PA2[Φ]. We allow Φ to be any collection of state-based, polynomial-time probabilistic Turing machines ϕ. The PA2[Φ] model is similar to the PA1 model, but is augmented with an encapsulation oracle that, when given some query information aux, computes $(C, K) = Encap(pk)$ and returns $(C, \phi(K, aux))$ to the ciphertext creator. We will only be interested in the case where Φ is the set of functions

$$\phi(K, aux) = \text{Enc}_K(\mathcal{P}(aux)) \qquad (13)$$

where $\mathcal{P}$ is a plaintext creator. Hence, ϕ takes the place of the plaintext creator and the DEM encryption algorithm. We note that if Null is the Turing machine that terminates giving no output, then PA2[{Null}] security is equivalent to Stam's notion of PA2 security.

Consider a KEM $(Gen, Encap, Decap)$, and a ciphertext creator $\mathcal{A}$ with associated plaintext extractor $\mathcal{A}^*$. We define PA2[Φ] using **REAL** and **FAKE** games. Let ϕ be any function in Φ. The **REAL** game is defined as follows:

1. The challenger generates a random key pair $(pk, sk) = Gen(1^k)$ and creates an empty list of encapsulations CList.
2. The attacker executes $\mathcal{A}$ on pk.
   - If the attacker queries the encryption oracle with query information aux, then the challenger generates an encapsulation $(C, K) = Encap(pk)$ and computes $\phi(K, aux)$. It adds $(C, \phi(K, aux))$ to CList and returns this value to the ciphertext creator.
   - If the attacker queries the decapsulation oracle with a ciphertext C, then the decapsulation oracle returns $\mathcal{D}(sk, C)$. The attacker may not query the decapsulation oracle with any encapsulation C for which there exists a value α such that $(C, \alpha) \in$ CList.

   The attacker terminates by outputting a bitstring x.

The **FAKE** game is defined as:

1. The challenger generates a random key pair $(pk, sk) = \mathcal{G}(1^k)$ and creates an empty list of ciphertexts CList.
2. The attacker executes $\mathcal{A}$ on pk.
   - If the attacker queries the encryption oracle with query information aux, then the challenger generates an encapsulation $(C, K) = Encap(pk)$ and computes $\phi(K, aux)$. It adds $(C, \phi(K, aux))$ to CList and returns this value to the ciphertext creator.
   - If the attacker queries the decapsulation oracle with a ciphertext C, then the decapsulation oracle returns $\mathcal{A}^*(C, pk, R[\mathcal{A}], \text{CList})$. The attacker may not query the decapsulation oracle with any encapsulation C for which there exists a value α such that $(C, \alpha) \in$ CList.

   The attacker terminates by outputting a bitstring x.

Definition 12. *A KEM is said to be PA2[Φ] if, for all ciphertext creators $\mathcal{A}$, there exists a plaintext extractor $\mathcal{A}^*$ such that for all $\phi \in \Phi$ and polynomial time distinguishers Dist the advantage*

$$|Pr[Dist(x) = 1 | \mathcal{A} \text{ plays } \boldsymbol{REAL}] - |Pr[Dist(x) = 1 | \mathcal{A} \text{ plays } \boldsymbol{FAKE}]| \qquad (14)$$

*that Dist has in distinguishing whether $\mathcal{A}$ interacts with the **REAL** game or the **FAKE** game is negligible as a function of the security parameter.*

We will show that a PA2 KEM and an unforgeable DEM combine to form a PA2 encryption scheme. We may answer encryption queries using the function $\phi(K, aux) = \text{Enc}_K(\mathcal{P}(aux))$. We may answer decryption queries as we do in the PA1 case, *except* for decryption queries of the form (C_1, C_2) where C_1 is an encapsulation returned by the encapsulation oracle. In this case, we return $\perp$ as it should be impossible for the attacker to find a new DEM ciphertext C_2 as the DEM is unforgeable. However, in order to prove this, we require that the KEM generates random-looking symmetric keys whenever the encryption oracle is queried. This is a generalisation of the normal IND-CCA2 security criterion for a KEM. Thus, our approach is only useful in showing that a hybrid scheme is plaintext-aware, and not that it is IND-CCA2, as our approach already requires the use of an IND-CCA2 KEM and an IND-CCA DEM[2]. The generalised notion of IND-CCA2 security we require is given below.

Definition 13. *Consider the following game played between a challenger and an attacker $\mathcal{A}$:*

1. *The challenger generates a key pair $(pk, sk) = \mathcal{G}(1^k)$, prepares an empty list of ciphertexts* CLIST*, and chooses a random bit $b \in \{0, 1\}$.*
2. *The attacker executes $\mathcal{A}$ on pk. The attacker has access to two oracles.*
   - *If the attacker queries the encapsulation oracle, then the challenger computes $(C, K_0) = Encap(pk)$. It also computes a random key K_1 of the same length as K_0, adds C to* CLIST *and returns (C, K_b) to the attacker.*
   - *If the attacker queries the decapsulation on C, then the challenger returns $Decap(sk, C)$. The attacker may not query the decapsulation oracle on any ciphertext $C \in$* CLIST*.*

   The attacker terminates by outputting a guess b' for b.

A KEM is said to be IND-CCA+ secure if every polynomial-time attacker's advantage

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

is negligible as a function of the security parameter.

Lemma 2. *A KEM is IND-CCA2 if and only if it is IND-CCA+.*

We may now prove the main theorem of this section.

Theorem 9. *Suppose that a KEM-DEM encryption scheme is composed of an INT-CCA+ secure DEM* (ENC, DEC) *and a KEM $(Gen, Encap, Decap)$ that is both IND-CCA+ and PA2[Φ] secure, where Φ is the set of functions*

$$\phi(K, aux) = \text{ENC}_K(\mathcal{P}(aux)) \tag{16}$$

and $\mathcal{P}$ is any plaintext creator. Then the encryption scheme is PA2.

Sketch Proof We will use standard game-hopping techniques. Let $\mathcal{A}$ be a ciphertext creator for the hybrid encryption scheme, and ϕ be any function in Φ. Let *Dist* be any polynomial-time distinguisher for $\mathcal{A}$ and let **Game 1** be similar to the **REAL** game, but where decryption oracle queries are responded to as follows.

[2] It may not be immediately obvious why the DEM must be IND-CCA secure. If we are using plaintext awareness to prove the security of a scheme, then we still require that the scheme is IND-CPA secure, which means that the DEM must be at least IND-PA secure. The conditions of the theorem which allow the combination of a secure KEM and a secure DEM to give a plaintext-aware encryption scheme insist that the DEM be INT-CCA+ secure. Any DEM that is IND-CPA and INT-CCA+ secure must be IND-CCA secure.

– If $\mathcal{A}$ queries the decryption oracle with a ciphertext (C_1, C_2), then check whether $(C_1, C_2') \in$ CList for some C_2'. If so, return $\perp$ to $\mathcal{A}$. Otherwise, return $\mathcal{D}(sk, (C_1, C_2))$.

Let $W_{0,Dist}$ is the event that $Dist(x) = 1$ when $\mathcal{A}$ interacts in the **REAL** game and $W_{1,Dist}$ is the event that $Dist(x) = 1$ when $\mathcal{A}$ interacts with decryption oracle in **Game 1**. Then there exists an attacker $\mathcal{A}_1$ for the IND-CCA+ game for the KEM with advantage ϵ_1 and an attacker $\mathcal{A}_2$ for the INT-CCA+ game for the DEM with success probability ϵ_2 such that

$$|Pr[W_{0,Dist}] - Pr[W_{1,Dist}]| \leq 4\epsilon_1 + \epsilon_2 \,. \tag{17}$$

This result is obtained by performing three separate game-hops. First we change the way the encryption and decryption oracles work so that random keys are used to encrypt and decrypt ciphertexts for which the encapsulation is the response from the encryption oracle. Next we always respond to decryption queries where the encapsulation is the same as in a response from the encryption oracle by outputting $\perp$, noting that if this is not the correct response, then the attacker has forged a new DEM ciphertext. Lastly, we change the way the encryption oracle works back to using the correct keys computed by the KEM, rather than with random keys.

Now, we define the ciphertext creator $\bar{\mathcal{A}}$ for the KEM as the algorithm that executes $\mathcal{A}$. We will assume, without loss of generality, that $\bar{\mathcal{A}}$ maintains a copy of CList. During $\mathcal{A}$'s execution:

– If $\mathcal{A}$ queries the decryption oracle with a ciphertext (C_1, C_2), then $\bar{\mathcal{A}}$ checks whether $(C_1, C_2') \in$ CList for some value of C_2'. If so, $\bar{\mathcal{A}}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\bar{\mathcal{A}}$ queries the encapsulation oracle with C_1. If the oracle returns $\perp$, then $\bar{\mathcal{A}}$ returns $\perp$ to $\mathcal{A}$. Otherwise the oracle returns a symmetric key K, and $\bar{\mathcal{A}}$ returns $\text{Dec}_K(C_2)$ to $\mathcal{A}$.
– If $\mathcal{A}$ queries the encryption oracle, then $\bar{\mathcal{A}}$ passes the query directly to the encapsulation oracle of the KEM and returns the result.

Since $\bar{\mathcal{A}}$ is a valid ciphertext creator for the KEM, there exists a plaintext extractor $\bar{\mathcal{A}}^*$ and a negligible function ϵ_3, such that:

$$|Pr[Dist(x) = 1|\bar{\mathcal{A}} \text{ plays } \textbf{REAL}] - |Pr[Dist(x) = 1|\bar{\mathcal{A}} \text{ plays } \textbf{FAKE}]| \leq \epsilon_3 \,. \tag{18}$$

We define a plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$ as follows. On the submission of a ciphertext (C_1, C_2), $\mathcal{A}^*$ checks whether $(C_1, C_2') \in$ CList for some value of C_2'. If so, $\mathcal{A}^*$ returns $\perp$. Otherwise, $\mathcal{A}^*$ executes $\bar{\mathcal{A}}^*$ on C_1. If $\bar{\mathcal{A}}^*$ returns $\perp$, then $\mathcal{A}^*$ returns $\perp$ to $\mathcal{A}$. If $\bar{\mathcal{A}}^*$ does not return $\perp$, then it must return a key K, and $\mathcal{A}^*$ returns $\text{Dec}_K(C_2)$ to $\mathcal{A}$. Let $W_{2,Dist}$ be the event that $Dist(x) = 1$ when $\mathcal{A}$ interacts with $\mathcal{A}^*$ instead of the proper decryption oracle.

It is easy to see that $\mathcal{A}$ produces output x in **Game 1** if and only if $\bar{\mathcal{A}}$ produces output x in **REAL**. Similarly, $\mathcal{A}$ produces output x when interacting with $\mathcal{A}^*$ (i.e. in the **FAKE** game) if and only if $\bar{\mathcal{A}}$ produces output x when interacting with $\bar{\mathcal{A}}^*$ (i.e. in the **FAKE** game). Therefore $|Pr[W_{1,Dist}] - Pr[W_{2,Dist}]| \leq \epsilon_3$ and so

$$|Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{REAL}] - |Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{FAKE}]| \leq 4\epsilon_1 + \epsilon_2 + \epsilon_3 \,. \tag{19}$$

□

4.4 Plaintext Awareness and IND-CCA2 KEMs

As a final note, we remark that the PA2[Φ] concept can also be used to prove the IND-CCA2 security in the standard model. If we let

$$\phi_1(K, aux) = K \tag{20}$$

and

$$\phi_2(K, aux) = K' \tag{21}$$

where K' is a randomly generated bit-string of the same length as K, and set $\Phi = \{\phi_1, \phi_2\}$, then a KEM that is IND-CPA and PA2[Φ] secure is necessarily IND-CCA2 secure (in the standard model). This generalises Stam's earlier result, given as Theorem 8. The proof of this can easily be adapted from Stam's work [16].

5 The Cramer-Shoup Scheme

In this section we will show that the Cramer-Shoup scheme, when applied to fixed length messages, is fully plaintext aware (PA2). This will prove a conjecture of Bellare and Palacio [4] by showing PA2 schemes can exist in the standard model. For our purposes, the Cramer-Shoup scheme will consist of the Cramer-Shoup KEM and an Encrypt-then-MAC DEM using a suitably secure encryption algorithm and MAC algorithm. Note that this is slightly different to the Cramer-Shoup scheme proven PA1 plaintext aware by Bellare and Palacio [4], but that similar techniques could have been used to prove that this scheme is PA1. We will define the Cramer-Shoup KEM as working over an arbitrary group G: this will make it easier to separate the properties required from the scheme from those that are required from the group.

Definition 14 (Cramer-Shoup KEM). *The Cramer-Shoup KEM is defined by the following three algorithms:*

- *The key generation algorithm which runs as follows:*
  1. *Generate a cyclic group G of order q and a generator g for G. Typically this will be either a subgroup of the finite field $GF(p)$ or a suitable elliptic curve group.*
  2. *Randomly select $w \in \mathbb{Z}_q^*$ and set $W = g^w$.*
  3. *Randomly select elements x, y and z from $\mathbb{Z}_q$, and set $X = g^x$, $Y = g^y$, and $Z = g^z$.*

4. *The public key consists of (g, W, X, Y, Z). The private key consists of (g, w, x, y, z). Note that both the encapsulation and decapsulation algorithms also make use of a hash function $Hash : G \times G \to \mathbb{Z}_q$ and a key derivation function $KDF : G \times G \to \{0,1\}^n$, where n is the (fixed) length of the required symmetric key.*

- *The encapsulation algorithm which runs as follows:*
  1. *Randomly select $u \in \mathbb{Z}_q$ and set $A = g^u$, $\hat{A} = W^u$ and $B = Z^u$.*
  2. *Set $K = KDF(A, B)$.*
  3. *Set $v = Hash(A, \hat{A})$.*
  4. *Set $D = X^u Y^{uv}$.*
  5. *Output the key K and the encapsulation $(A, \hat{A}, D)$.*
- *The decapsulation algorithm which runs as follows:*
  1. *Set $v = Hash(A, \hat{A})$.*
  2. *Check that $D = A^{x+yv}$ and that $\hat{A} = A^w$. If not, output $\perp$ and halt.*
  3. *Otherwise, set $B = A^z$.*
  4. *Output $K = KDF(A, B)$.*

5.1 Cramer-Shoup is Simulatable

In order to show that the Cramer-Shoup scheme is PA2, we need to show two separate things: that it is PA1+ and that it is simulatable. In this section we will show that the Cramer-Shoup scheme is simulatable. In order to do this we have to show that we can find Turing machines f and f^{-1} that satisfy Definition 6. Owing to the results of Section 3.4, it is enough to show that there exists Turing machines (Kf, Kf^{-1}) and (Df, Df^{-1}) that simulate the KEM and DEM respectively.

We construct our DEM from a suitably secure block cipher running in counter mode and from the EMAC MAC algorithm. Details of both of these schemes can be found in, for example, [9].

Theorem 10. *An Encrypt-then-MAC DEM composed of the counter mode encryption scheme and the EMAC MAC algorithm is simulatable if the underlying block cipher is indistinguishable from random.*

Sketch Proof First, we note that the decryption oracle to which the attacker has access is of no use due to the unforgeability of the MAC. Hence, we remove it. The result then follows from the indistinguishability of the MAC code [14] and the indistinguishability of the counter mode encryption [1]. □

We will now show that the Cramer-Shoup KEM is simulatable providing that it is instantiated on a group that is simulatable.

Definition 15 (Simulatable Group). *A group G is simulatable if there exist two polynomial-time Turing machines (f, f^{-1}) such that:*

- *f is a deterministic Turing machine that takes elements $r \in \{0,1\}^l$ as input, and outputs elements of G.*

– f^{-1} *is a probabilistic Turing machine that takes elements of* $h \in G$ *as input, and outputs elements of* $\{0,1\}^l$.
– $f(f^{-1}(h)) = h$ *for all* $h \in G$.
– *There exists no polynomial-time attacker* $\mathcal{A}$ *that has a non-negligible advantage in winning the following game:*
  1. *The challenger randomly chooses a bit* $b \in \{0,1\}$.
  2. *The attacker executes* $\mathcal{A}$ *on the input* 1^k. *The attacker has access to an oracle* $\mathcal{O}_f$ *that takes no input, generates a random element* $r \in \{0,1\}^l$, *and returns* r *if* $b = 0$ *and* $f^{-1}(f(r))$ *if* $b = 1$. *The attacker terminates by output a guess* b' *for* b.

  The attacker wins if $b = b'$ *and its advantage is defined in the usual way.*
– *There exists no polynomial-time attacker* $\mathcal{A}$ *that has a non-negligible advantage in winning the following game:*
  1. *The challenger randomly chooses a bit* $b \in \{0,1\}$.
  2. *The attacker executes* $\mathcal{A}$ *on the input* 1^k. *The attacker has access to an oracle* $\mathcal{O}_f$ *that takes no input. If* $b = 0$, *then the oracle generates a random* $r \in \{0,1\}^l$ *and returns* $f(r)$. *Otherwise the oracle generates a random* $h \in G$ *and returns* h. *The attacker terminates by outputting a guess* b' *for* b.

  The attacker wins if $b = b'$ *and its advantage is defined in the usual way.*

Theorem 11. *The Cramer-Shoup KEM is simulatable if it is instantiated on a simulatable group* G *on which the DDH problem is hard, and under the assumptions that the hash function Hash is target collision resistant and that the key derivation function KDF is unpredictable with random inputs.*

These assumptions are formally defined as follows. The notation is taken from the Cramer and Shoup paper [6].

Definition 16 (DDH). *For any polynomial-time algorithm* $\mathcal{A}$ *that outputs a single bit, we define* $AdvDDH$ *to be*

$$\begin{aligned} &|Pr[\mathcal{A}(g, g^x, g^y, g^{xy}) = 1 | x, y \text{ chosen randomly from } \mathbb{Z}_q] \\ &\quad -Pr[\mathcal{A}(g, g^x, g^y, g^z) = 1 | x, y, z \text{ chosen randomly from } \mathbb{Z}_q]| \end{aligned} \tag{22}$$

The DDH assumption is that, for all polynomial-time algorithms $\mathcal{A}$, $AdvDDH$ *is negligible as a function of the security parameter.*

Definition 17 (TCR). *Let Hash be the hash function used within the Cramer-Shoup scheme. For any polynomial-time algorithm* $\mathcal{A}$, *we define* $AdvTCR$ *to be*

$$\begin{aligned} &Pr[\mathcal{A}(\phi^*) \neq \phi^* \wedge Hash(\mathcal{A}(\phi^*)) = Hash(\phi^*) \\ &\qquad | \phi^* \text{ chosen randomly from } \langle g \rangle \times \langle g \rangle] \end{aligned} \tag{23}$$

The target collision resistance (TCR) assumption is that, for all polynomial-time algorithms $\mathcal{A}$, $AdvTCR$ *is negligible as a function of the security parameter.*

Definition 18 (KDF). *Let KDF be the key derivation function used within the Cramer-Shoup scheme and l be the length of symmetric keys that the scheme is required to produce. Let E_1 be the event that A and B are chosen randomly from $\langle g \rangle$ and E_2 be the event that A is chosen randomly from $\langle g \rangle$ and K is chosen randomly from $\{0,1\}^n$. For any polynomial time algorithm $\mathcal{A}$ that outputs a single bit, we define $AdvDist(KDF)$ to be*

$$|Pr[\mathcal{A}(g, A, KDF(A, B)) = 1 | E_1] - Pr[\mathcal{A}(g, A, K) = 1 | E_2]| \tag{24}$$

The distribution assumption for KDF is that, for all polynomial-time algorithms $\mathcal{A}$, $AdvDist(KDF)$ is negligible as a function of the security parameter.

Proof of Theorem 11 Let $\mathcal{A}$ be any attacker that is attempting to distinguish a real encapsulation pair (C, K) from a simulated encapsulation $(f(r), K')$ where r is a randomly generated bitstring of length l and K' is a randomly chosen symmetric key of the appropriate length. We will assume $\mathcal{A}$ makes at most q_E encapsulation oracle queries and q_D decapsulation oracle queries. Let **Game 1** be the game in which interacts with correct encapsulation and decapsulation oracles. Let **Game 2** be the game in which, for its first query to the encapsulation oracle, the attacker is interacting with the following algorithm rather than the true encapsulation algorithm:

1. Randomly select $u \in \mathbb{Z}_q$ and set $A = g^u$.
2. Randomly select $\hat{u} \in \mathbb{Z}_q \setminus \{u\}$ and set $\hat{A} = g^{\hat{u}}$.
3. Randomly select $K \in \{0,1\}^n$.
4. Set $v = Hash(A, \hat{A})$ and $D = X^u Y^{uv}$.
5. Output the encapsulation $(A, \hat{A}, D)$ and the symmetric key K.

Let W_i be the event that the attacker $\mathcal{A}$ wins Game i. We use a result of Cramer-Shoup [6] to take us most of the way towards our goal.

Lemma 3 (Cramer-Shoup).

$$|Pr[W_1] - Pr[W_2]| \leq AdvDDH + AdvTCR + AdvDist(KDF) + (q_E + 3)/q \tag{25}$$

Let **Game 3** be the game in which $\hat{A}$ is computed as follows:

2. Randomly select $\hat{u} \in \mathbb{Z}_q$ and set $\hat{A} = g^{\hat{u}}$.

Clearly the two games are identical unless $\hat{u} = u$, hence:

$$|Pr[W_2] - Pr[W_3]| \leq 1/q\,. \tag{26}$$

Let **Game 4** be the game in which D is computed as follows:

4. Randomly select $r' \in \mathbb{Z}_q$ and set $D = g^{r'} Y^{uv}$.

Clearly, any difference in behaviour of the attacker between Game 3 and Game 4 means that he has distinguished between the Diffie-Hellman triple (A, X, X^u) and $(A, X, g^{r'})$. [Note that the proof makes use of the fact that we may compute Y^{uv} as A^{vy} in the case that we know y but do not know the discrete logarithm of A.] Hence,

$$|Pr[W_3] - Pr[W_4]| \leq AdvDDH\,. \tag{27}$$

Let **Game 5** be the game in which D is computed as follows:

4. Randomly select $r' \in \mathbb{Z}_q$ and set $D = g^{r'}$.

This difference is pure conceptual, and so $Pr[W_4] = Pr[W_5]$. However, now each of the elements of the ciphertext, and the symmetric key, are randomly generated from their appropriate ranges. At this stage, and merely through altering the way we respond to the first encryption oracle query, we have

$$|Pr[W_1]-Pr[W_5]| \leq 2\cdot AdvDDH+AdvTCR+AdvDist(KDF)+(q_E+4)/q\,. \tag{28}$$

Let **Game 6** be the game in which each of the encapsulation oracle queries is answered using the algorithm in Game 5, and not just the first one. By repeated application of the previous results we have that:

$$|Pr[W_1]-Pr[W_6]| \leq q_E\big\{2\cdot AdvDDH+AdvTCR+AdvDist(KDF))+(q_E+4)/q\big\}\,. \tag{29}$$

Note that, at this stage, every single group element returned by the encryption oracle is randomly generated.

Lastly, suppose the group G can be simulated by the pair of Turing machines (Gf, Gf^{-1}), and let **Game 7** be the game in which the encapsulation oracle computes the ciphertexts as follows:

1. Randomly select $r_1 \in \{0,1\}^l$ and set $A = Gf(r_1)$.
2. Randomly select $r_2 \in \{0,1\}^l$ and set $\hat{A} = Gf(r_2)$.
3. Randomly select $r_3 \in \{0,1\}^l$ and set $D = Gf(r_3)$.
4. Randomly select $K \in \{0,1\}^n$.
5. Output the encapsulation $(A, \hat{A}, D)$ and the symmetric key K.

Since the group is simulatable, the difference between success probabilities when the encapsulation is provided as in Game 6 and in Game 7 is negligible. However this means that the difference between $Pr[W_1]$ and $Pr[W_7]$ is negligible, and so the KEM is simulatable. $\square$

Lastly, we wish to show that simulatable groups exist. The obvious method to attempt to simulate a cyclic group G of order q with generator g is to define

$$f : \{0,1\}^l \to G \quad \text{by setting} \quad f(r) = g^r \tag{30}$$

where $l \gg q$. This provides a perfectly adequate definition of f, but leaves us know way of computing a machine f^{-1} (without solving the discrete logarithm problem in G!). We are therefore required to use sneakier techniques.

Lemma 4. *If q and p are primes such that $p = 2q + 1$, and G is the subgroup of $\mathbb{Z}_p^*$ of order q, then G is simulatable.*

Proof Suppose we let l be a large number, say $\alpha(\lceil \log_2(p) \rceil + \beta)$ for some integer values α and β, and define f to work as follows:

1. Split the input $x \in \{0, 1\}^l$ into α blocks $x_1, \ldots, x_\alpha$ of length $\lceil \log_2(p) \rceil + \beta$.
2. Compute $X_1 = x_1 \bmod p$.
3. Test whether $X_1 \in G$. If so, output X_1 and halt. Otherwise repeat with the next input block.

For large enough values of β the value of each X_i will be (approximately) uniformly distributed over $\mathbb{Z}_p$. Furthermore, this algorithm will only fail to output an (almost) randomly selected element of G if none of the X_i values are in G. The probability that this occurs is bounded above by $1/2^\alpha$ as $p = 2q + 1$. This value is negligible if, for example, $\alpha = k$.

We also need to define f^{-1}. Let g be a generator of G. For an input $g^x \in G$, f^{-1} can be computed as follows:

1. For each i with $1 \leq i \leq \alpha$, randomly choose a bit $b \in \{0, 1\}$.
2. If $b = 0$ then:
   (a) Randomly select a bit string x_i of length $\lceil \log_2(p) \rceil + \beta$.
   (b) Compute $X_i = x_i \bmod p$.
   (c) Test whether $X_i \in G$. If not, output x_i and continue the algorithm.
   (d) Otherwise choose another bit string x_i and start the subroutine again.
3. If $b = 1$ then:
   (a) Randomly select a bit string x_i of length $\lceil \log_2(p) \rceil + \beta$.
   (b) Compute $X_i = x_i \bmod p$.
   (c) Compute $\delta = X_i - g^x \bmod p$.
   (d) Let Δ be the bit string of length $\lceil \log_2(p) \rceil + \beta$ that represents δ. If $x_i + \Delta$ (considered as an integer) is greater than $2^{\lceil \log_2(p) \rceil + \beta}$ then randomly select a new bit string x_i and start the subroutine again.
   (e) Otherwise, compute $x_i' = x_i + \Delta$.
   (f) Output x_i'.
   (g) Output $(\alpha - i)(\lceil \log_2(p) \rceil + \beta)$ random bits and terminate the whole algorithm

We will allow the inner-loops to run at most k times. Each inner loop has a failure probability that is bounded above by $1/2 + 1/2p$. Therefore the probability that the inner loop fails to produce an output is negligible. Similarly, the function fails to produce a valid output if the bit $b = 0$ is continually selected. This occurs with probability $1/2^\alpha$, which is negligible if, for example, $\alpha = k$. $\square$

5.2 Cramer-Shoup is PA1+

Now we are only require to show that the Cramer-Shoup KEM is PA1+ to complete our proof that the Cramer-Shoup scheme is PA2. In this section we show that Cramer-Shoup is PA1+ on a simulatable group under the DHK assumption.

The DHK assumption states that any attacker given a random element W in a group generated by g, can only compute a Diffie-Hellman triple (W, g^u, W^u) if they know u.

Definition 19 (DHK). *Let G be a cyclic group G of order q and a generator g for G. The DHK assumption for G is that for any polynomial-time algorithm $\mathcal{A}$ there exists a polynomial-time extractor $\mathcal{A}^*$ such that the probability that $\mathcal{A}$ wins the following game is negligible.*

1. *The challenger randomly chooses an element $W \in G$.*
2. *The attacker executes $\mathcal{A}$ on the input W. The attacker has access to an oracle which, when given a triple $(W, A, \hat{A}) \in G^3$, executes $\mathcal{A}^*(W, A, \hat{A}, R[\mathcal{A}])$ and returns the result.*

The attacker wins the game if it submits a triple of the form (W, g^u, W^u) to the oracle and the oracle fails to return u. The challenger wins the game if $\mathcal{A}$ terminates without this event occurring.

The DHK assumption is certainly a very strong one. It was essentially introduced by Damgård in 1991 [7] and has been used in a number of applications [3, 4, 11, 12]. However, it is unclear if the assumption holds true or not. Opponents of the assumption point out that it is not falsifiable (and so demonstrations that it is false must be complex) [13] and that variants of the assumption have been proven false [3]. Nevertheless, it is used to prove that a version of the Cramer-Shoup scheme is PA1 [4] and so we consider it a reasonable assumption under which to prove that the Cramer-Shoup scheme is PA2. The question of whether plaintext awareness can be demonstrated under weaker assumptions is a major open problem.

Theorem 12. *The Cramer-Shoup KEM is PA1+ in a simulatable group under the DHK assumption*

Sketch Proof Let $\mathcal{A}$ be any PA1+ ciphertext creator. We use the assumption that we can find algorithms that solve the DHK problem to build a plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$.

Consider the following plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$ that makes use of a DHK oracle. When it is first invoked, $\mathcal{A}^*$ receives the public key (W, X, Y, Z) and the random coins $R[\mathcal{A}]$ of $\mathcal{A}$. It first simulates the random coins of an attacker that only received W from the challenger and computed X, Y and Z. This is necessary because the DHK assumption is only valid when the challenger gives the attacker a single group element W. The simulated random coins string is given by:

$$R = Gf^{-1}(X)||Gf^{-1}(Y)||Gf^{-1}(Z)||R[\mathcal{A}] \tag{31}$$

where Gf^{-1} is the inverse function associated with the simulatable group. If $\mathcal{A}$ makes a decryption oracle query on the ciphertext $(A, \hat{A}, D)$ then $\mathcal{A}^*$ proceeds as follows:

1. Query the DHK oracle with the triple $(W, A, \hat{A})$ and the coins (R, RList). The oracle will return a value $u \in \mathbb{Z}_q$ or the error symbol $\perp$. If the oracle returns $\perp$, then return $\perp$ and terminate.
2. Set $v = Hash(A, \hat{A})$.
3. Check that $A = g^u$, $\hat{A} = W^u$ and $D = X^u Y^{uv}$. If not, return $\perp$.
4. Set $B = Z^u$.
5. Set $K = KDF(A, B)$.
6. Return K.

It is clear that $\mathcal{A}^*$ correctly simulates the decapsulation algorithm providing that it obtains correct solutions to the DHK problem from the DHK oracle. The DHK assumption states that there exists an algorithm $\mathcal{A}'$ that can answer the queries of the DHK oracle given the randomness that $\mathcal{A}$ used in creating these queries. It is important to note that because the DHK oracle must give back answers which are completely correct, *and not answers that are merely indistinguishable from correct by* $\mathcal{A}$, it is sufficient to give $\mathcal{A}'$ access to the random coins that $\mathcal{A}$ used in creating its challenge. In other words, it is sufficient for $\mathcal{A}'$ to take as input the random coins R and all the random blocks RList that have been received by $\mathcal{A}$ up to the point at which the DHK oracle query was made. Hence, by the DHK assumption, there exists an algorithm $\mathcal{A}'$ that correctly responds to the DHK oracles queries, and so there exists a plaintext extractor $\mathcal{A}^*$ for $\mathcal{A}$. Hence, the Cramer-Shoup KEM is PA1+. □

6 Conclusion

We have shown that the Cramer-Shoup scheme is PA2 plaintext aware and therefore demonstrated the existence of fully plaintext aware encryption algorithms. However, in order to do this, we have had to use results which demonstrate that the Cramer-Shoup scheme is IND-CCA2 secure already. Therefore, if the primary goal of plaintext awareness is to make proving the security of an encryption scheme easier, then the results of this paper are of little use. We present these results not as a practical tool, but as a proof that PA2 plaintext aware schemes can be shown to exist.

We have also demonstrated sufficient conditions that a KEM and a DEM must fulfil if the resulting hybrid encryption scheme is plaintext aware.

Acknowledgements

The author would like to thank Martijn Stam of the University of Bristol for his detailed and insightful comments on the several drafts of this paper. In particular, Theorem 2 of Section 3.1 and the results of Section 4.4 are the direct result of his comments and questions. Thanks should also be given to Nigel Smart for his comments and to the anonymous referees of Eurocrypt 2006. The author also gratefully acknowledges the financial support of the EPSRC and the European Union's ECRYPT project.

References

1. 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.
2. 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.
3. M. Bellare and A. Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In M. Franklin, editor, *Advances in Cryptology – Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 273–289. Springer-Verlag, 2004.
4. M. Bellare and A. Palacio. Towards plaintext-aware public-key encryption without random oracles. In P. J. Lee, editor, *Advances in Cryptology – Asiacrypt 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 48–62. Springer-Verlag, 2004.
5. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology – Eurocrypt '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer-Verlag, 1994.
6. 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.
7. I. B. Damård. Towards practical public key systems secure against chosen ciphertext attacks. In J. Feigenbaum, editor, *Advances in Cryptology – Crypto '91*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456. Springer-Verlarg, 1991.
8. A. W. Dent. Hybrid cryptography. Available from `http://eprint.iacr.org/2004/210/`, 2004.
9. A. W. Dent and C. J. Mitchell. *User's Guide to Cryptography and Standards.* Artech House, 2005.
10. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the RSA assumption. In J. Kilian, editor, *Advances in Cryptology – Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 260–274. Springer-Verlag, 2001.
11. S. Hada and T. Tanaka. On the existence of 3-round zero-knowledge protocols. In H. Krawcyzk, editor, *Advances in Cryptology – Crypto '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 408–423. Springer-Verlag, 1998.
12. H. Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In V. Shoup, editor, *Advances in Cryptology – Crypto 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 546–566. Springer-Verlag, 2005.
13. M. Naor. On cryptographic assumptions and challenges. In D. Boneh, editor, *Advances in Cryptology – Crypto 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer-Verlag, 2003.
14. E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *Journal of Cryptography*, 13(3):315–339, 2000.
15. V. Shoup. OAEP reconsidered. In J. Kilian, editor, *Advances in Cryptology – Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 239–259. Springer-Verlag, 2001.

16. M. Stam. A key encapsulation mechanism for NTRU. In N. P. Smart, editor, *Coding and Cryptography, Proc. 10th IMA International Conference*, volume 3796 of *Lecture Notes in Computer Science*, pages 410–427. Springer-Verlag, 2005.

A Major Differences from Version 1

This is the second version of this paper to appear on the web, and since the focus of the paper has changed significantly since the first version was published, I thought it might be helpful to point the reader to these changes.

– **Focus.** The original focus of this paper was on the criteria that a KEM and DEM should fulfil in order that the overall hybrid encryption scheme be plaintext aware. In particular, much was made of the results of Section 4.3. However, since the results on encryption simulation seem likely to have a greater bearing on future research, the focus of the paper has been changed so that these results are highlighted more prominently.
– **A false claim was corrected.** The original version of this paper claimed that a simulatable and PA1 encryption scheme was PA2. Subsequently an error was found in the proof, and the paper was withdrawn. This version presents a slightly weaker result, that a simulatable and PA1+ encryption scheme is PA2, and proves that the Cramer-Shoup scheme is PA1+.
– **It has been noted that simulatability implies IND-CCA2 security.** This was noted by Martijn Stam, and pretty much kills off the idea that encryption simulation can be used to help prove that ECIES is IND-CCA2 secure in the standard model.
– **The criteria for a KEM to be PA1 is now necessary as well as sufficient.** Theorem 6 was not in the original version of the paper. This provides an equivalence between PA1 KEMs and hybrid encryption schemes that are PA1 regardless of the DEM used. It does not say anything about schemes that are PA1 with a particular DEM.
– **The relationship between PA2[Φ] and IND-CCA2 KEMs has been corrected.** The results of Section 4.4 were originally incorrect. Thanks to Martijn Stam for pointing this out to me.

B Major Differences from Version 2

– **An error in the proof that Cramer-Shoup is PA1+ has been corrected**. The original version of the proof did not require the group to be simulatable. This meant it was impossible to equate the extra group elements of the public key with random strings.