The Cramer-Shoup Encryption Scheme is Plaintext Aware in the Standard Model

Alexander W. Dent

Royal Holloway, University of London
Egham, Surrey, TW20 0EX, U.K.
a.dent@rhul.ac.uk

Abstract. In this paper we examine the notion of plaintext awareness as it applies to hybrid encryption schemes. 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 fully 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 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, a definition of plaintext awareness has been given in the standard model [4], where the plaintext extractor is also 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 [5] was a very likely candidate.

This paper proves that the Cramer-Shoup scheme is plaintext aware in the standard model, thus proving the conjecture of Bellare and Palacio. The proof uses two new techniques: *encryption simulation* and *PA1+ plaintext awareness*. An encryption scheme that is simulatable is necessarily IND-CCA2 secure, and so the concept has limited use. However, the concept of PA1+ plaintext awareness may have further scope. The proof is obtained 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 [5]. 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 accepted notion of security for 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 a decryption oracle that, when given $C \in \mathcal{C}$ will return $\mathcal{D}(sk, C)$.
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 execution, $\mathcal{A}_2$ may query a decryption oracle, 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:

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

Definition 1. *If, for all polynomial-time attackers $\mathcal{A}$, the advantage that $\mathcal{A}$ has in winning the above game for an encryption scheme $(\mathcal{G}, \mathcal{E}, \mathcal{D})$ is negligible as a function of the security parameter k, then that encryption scheme is said to be IND-CCA2 secure.*

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

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 $K = \mathit{Decap}(sk, C)$.

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 $\perp$.

A KEM and a DEM can be composed in the obvious way in create a hybrid encryption algorithm. The greatest advantage of designing a hybrid encryption scheme in terms of KEMs and DEMs is that Cramer and Shoup [5] were able to propose independent security criteria for the KEM and the DEM that guarantee that a secure KEM and a secure DEM combine to give a secure (IND-CCA2) encryption scheme. However, since our focus is on plaintext awareness, we will not need to discuss these security notions here.

2.2 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 in the standard model 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 their ability to access the same random tape between invocations.

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 $\textsc{CList}$.
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 $\textsc{CList}$ and returns C to the attacker.

[1] Technically, the plaintext creator will only generate a random message, and it will be left to the challenge 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.

  – 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 2 (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*

$$|Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{\textit{REAL}}] - |Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{\textit{FAKE}}]| \quad (2)$$

*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

$$|Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{\textit{REAL}}] - |Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{\textit{FAKE}}]| \quad (3)$$

is negligible as a function of the security parameter.

3 Simulatable Encryption Schemes

The aim of this paper is to show that the 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 roughly 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. In this section we will formally define simulation and PA1+ plaintext awareness, and show that any scheme that is both PA1+ and simulatable is PA2.

3.1 Simulatable Encryption

We will wish to work with encryption schemes that are simulatable, by which we mean that there exists a 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 Turing machine f^{-1} which acts as a perfect inverse for f when used on the right, i.e.

$$f(f^{-1}(C)) = C \text{ for all } C \in \mathcal{C}\,. \tag{4}$$

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

Sketch 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:

$$\frac{1}{2}\left|Pr[\mathcal{B} \text{ outputs } 1|b=0] - Pr[\mathcal{B} \text{ outputs } 1|b=1]\right| . \quad (5)$$

However,

$$\left|Pr[\mathcal{B} \text{ outputs } 1|b=0] - Pr[\mathcal{B} \text{ outputs } 1|b=1]\right| = \left|Pr[W_0] - Pr[W_1]\right| . \quad (6)$$

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 our goal in this paper is to show that PA2 schemes can exist, this notion will prove useful.

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 roughly 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 $\mathcal{A}$ 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 RList of the random blocks that the attacker has been given.
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 $\mathcal{A}^*(C, pk, R[\mathcal{A}], \text{RList})$.

   The attacker terminates by outputting a bitstring x.

Definition 4 (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*

$$|Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{\textit{REAL}}] - |Pr[Dist(x) = 1|\mathcal{A} \text{ plays } \textbf{\textit{FAKE}}]| \qquad (7)$$

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.

Theorem 2. *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 PA2 ciphertext creator for the 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 Definition 3. 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 3. 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 $W_{5,Dist(x)}$ be the event that $Dist(x) = 1$ in this model. As before, if there is any significant difference between $Pr[W_{4,Dist(x)}]$ and $Pr[W_{5,Dist(x)}]$, then we may build an algorithm that distinguishes between ciphertexts and simulated ciphertexts, contravening Definition 3. Therefore,

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

is negligible. However, this means that

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

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. □

3.3 PA1+ Plaintext-Aware KEMs

It will be convenient for us to work with the hybrid version of the Cramer-Shoup encryption scheme. In this section we will 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 for a KEM is the obvious extension of the PA1+ model for an encryption scheme. 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 5. *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*

$$|Pr[Dist(x) = 1|\mathcal{A} \textit{ plays } \textbf{\textit{REAL}}] - |Pr[Dist(x) = 1|\mathcal{A} \textit{ plays } \textbf{\textit{FAKE}}]| \quad (8)$$

*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 3. *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 $\mathrm{DEC}_K(C_2)$ to $\mathcal{A}$. Any queries that $\mathcal{A}$ makes to the randomness oracle are passed directly on to $\bar{\mathcal{A}}$'s randomness oracle, and the results returned 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 $\mathrm{DEC}_K(C_2)$. 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. □

4 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 6 (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.*
  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, q, W, X, Y, Z). The private key consists of (g, q, 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 $K = KDF(A, B)$.*

4.1 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 3. It is enough to show that there exists Turing machines (Kf, Kf^{-1}) and (Df, Df^{-1}) that simulate the KEM and DEM respectively. The function Df must accurately simulate the encryption of a fixed-length message by the DEM under a random key. The function Kf should produce encapsulations for which it is impossible to distinguish a correct encapsulation pair (C, K) from a simulated encapsulation pair $(Kf(r), K')$, where r is a randomly generated bitstring of length l and K' is a randomly generated symmetric key of the appropriate length. Formal treatments are given in the full version of the paper.

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, [7].

Theorem 4. *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 [11] 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. Again, we only provide a loose description of a simulatable group here, leaving the formal description to the full version of this paper. A group is simulatable if there exists Turing machines (Gf, Gf^{-1}) analogous to those in Definition 3 for which it is impossible to distinguish between a randomly chosen group element $h \in G$ and a simulated group element $Gf(r)$, where r is chosen randomly from the set $\{0,1\}^l$.

Theorem 5. *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 [5].

Definition 7 (DDH). *For any polynomial-time algorithm $\mathcal{A}$ that outputs a single bit, we define $AdvDDH$ to be*

$$\begin{aligned}&|Pr[\mathcal{A}(p,q,g,g^x,g^y,g^{xy}) = 1 | x,y \text{ chosen randomly from } \mathbb{Z}_q] \\ &\quad - Pr[\mathcal{A}(p,q,g,g^x,g^y,g^z) = 1 | x,y,z \text{ chosen randomly from } \mathbb{Z}_q]| \qquad (9)\end{aligned}$$

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

Definition 8 (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*

$$\begin{aligned}&Pr[\mathcal{A}(\phi^*) \neq \phi^* \wedge Hash(\mathcal{A}(\phi^*)) = Hash(\phi^*) \\ &\qquad\qquad | \phi^* \text{ chosen randomly from } \langle g \rangle \times \langle g \rangle] \qquad (10)\end{aligned}$$

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 9 (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*

$$|Pr[\mathcal{A}(p,q,g,A,KDF(A,B)) = 1 | E_1] - Pr[\mathcal{A}(p,q,g,A,K) = 1 | E_2]| \qquad (11)$$

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 5 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 encryption and decryption 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 $v = \mathit{Hash}(A, \hat{A})$ 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 [5] to take us most of the way towards our goal.

Lemma 1 (Cramer-Shoup).

$$|Pr[W_1] - Pr[W_2]| \leq \mathit{AdvDDH} + \mathit{AdvTCR} + \mathit{AdvDist}(\mathit{KDF}) + (q_E + 3)/q \quad (12)$$

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:

$$|Pr[W_2] - Pr[W_3]| \leq 1/q\,. \quad (13)$$

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,

$$|Pr[W_3] - Pr[W_4]| \leq \mathit{AdvDDH}\,. \quad (14)$$

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

$$|Pr[W_1] - Pr[W_5]| \leq 2 \cdot \mathit{AdvDDH} + \mathit{AdvTCR} + \mathit{AdvDist}(\mathit{KDF}) + (q_E + 4)/q\,. \quad (15)$$

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:

$$|Pr[W_1]-Pr[W_6]| \leq q_E\left\{2\cdot AdvDDH+AdvTCR+AdvDist(KDF))+(q_E+4)/q\right\}. \tag{16}$$

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 is negligible. However this means that the difference between $Pr[W_1]$ and $Pr[W_7]$ is negligible, and so the KEM is simulatable. □

Lastly, we need 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

$$f : \{0,1\}^l \to G \quad \text{by setting} \quad f(r) = g^r \tag{17}$$

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.

Theorem 6. *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.*

Sketch Proof To show that G is simulatable, we are required to find Turing machines Gf and Gf^{-1} that are analogous to those given in Definition 3 but for which the output of Gf is a group element. Let k be an integer much larger than $\log_2(q)$ and let α be an integer. We consider a map $Gf : \{0,1\}^{\alpha k} \to G$ as follows. First, split the input r into α k-bit substrings $r = r_1||r_2||\ldots||r_\alpha$. Next, consider r_1 as an integer modulo p and test whether it is in G. If so, output $r_1 \bmod p$; otherwise consider the next substring of r in the same way. Since the distribution of $r_i \bmod p$ is almost uniform, we have that the probability that this algorithm fails to return a random element of G is approximately $1/2^\alpha$.

The inverse machine Gf^{-1} works similarly. Given a group element g, first chooses a random bit $b \in \{0,1\}$. If $b = 0$ then construct a random string r_1 of length k such that $r_1 \bmod p \equiv g$, append random data to r_1 so that it is αk-bits long and output the result. If b=1, then construct a random element r_1 such that $r_1 \bmod p \notin G$ and choose a new random bit for the block r_2. This process continues until either we choose a bit $b = 0$ or we have constructed α blocks of data (at which point the algorithm fails). Again, the probability that this happens is approximately $1/2^\alpha$. □

4.2 Cramer-Shoup is PA1+

Now we are only require to show that the Cramer-Shoup scheme is PA1+ to complete our proof that it 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 10 (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 [6] and has been used in a number of applications [3, 4, 8, 9]. 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) [10] 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 7. *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:

$$R = Gf^{-1}(X)||Gf^{-1}(Y)||Gf^{-1}(Z)||R[\mathcal{A}] \tag{18}$$

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+. □

5 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.

Acknowledgements

The author would like to thank Martijn Stam for his detailed and insightful comments on the several drafts of this paper. Thanks should also be given to both Nigel Smart and the anonymous referees for their helpful comments. The author gratefully acknowledges the financial support of the EPSRC.

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. R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2004.
6. 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.
7. A. W. Dent and C. J. Mitchell. *User's Guide to Cryptography and Standards.* Artech House, 2005.
8. 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.
9. 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.
10. 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.
11. E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *Journal of Cryptography*, 13(3):315–339, 2000.