Efficient KEMs with Partial Message Recovery

T.E. Bjørstad[1], A.W. Dent[2], and N.P. Smart[3]

[1] The Selmer Center, Department of Informatics,
University of Bergen, Pb. 7800, N-5020 Bergen, Norway.
Email : tor.bjorstad@ii.uib.no
[2] Information Security Group, Royal Holloway, University of London,
Egham, Surrey, TW20 0EX, United Kingdom.
Email: a.dent@rhul.ac.uk
[3] Department Computer Science, University of Bristol,
Merchant Venturers Building, Woodland Road,
Bristol, BS8 1UB, United Kingdom.
Email: nigel@cs.bris.ac.uk

Abstract. Constructing efficient and secure encryption schemes is an important motivation for modern cryptographic research. We propose simple and secure constructions of hybrid encryption schemes that aim to keep message expansion to a minimum, in particular for RSA-based protocols. We show that one can encrypt using RSA a message of length $|m|$ bits, at a security level equivalent to a block cipher of κ bits in security, in $|m| + 4\kappa + 2$ bits. This is therefore independent of how large the RSA key length grows as a function of κ. Our constructions are natural and highly practical, but do not appear to have been given any previous formal treatment.

1 Introduction

There are many important factors to consider when choosing a practical encryption scheme, including speed, provable security, code size, and bandwidth efficiency. Bandwidth efficiency is often only considered as an afterthought, but for many real world problems this can be as important as keeping the computational and implementational complexity low. In particular this is the case for wireless settings, where power consumption often is a limiting factor, and transmitting data is a major power drain compared to the cost of doing some extra (offline) computation [18].

Suppose that we wish to encrypt $|m|$-bit messages with a security level of κ bits, for example $\kappa = 128$. The aim of this paper is to suggest secure protocols which achieves the desired security level, while keeping the length of the ciphertexts as close to $|m|$ bits as possible. The existing ISO/IEC standard for public-key encryption [17] only considers bandwidth efficiency for short messages and does not give an RSA-based scheme that is efficient for arbitrary-length messages and arbitrary RSA key sizes.

To explain our starting point in more concrete terms, let R_κ denote the size of an RSA key equivalent to κ bits of security, and similarly let E_κ be the

corresponding key size for elliptic curve based systems. At the time of writing, it is widely believed that approximate values for these parameters are roughly $R_{80} = 1024$, $R_{128} = 3072$, and $E_\kappa = 2\kappa$ [14].

We first examine the problem by analogy with digital signature schemes. Using the ECDSA signature scheme one obtains a short signature scheme with appendix, of total length $|m| + 2E_\kappa$ bits, that is, in addition to the original message we get a signature appendix consisting of two group elements. With pairings, it is possible to shrink the size of signatures to $|m| + E_\kappa$ bits using the BLS scheme [9]. Standard RSA-based schemes such as RSA-FDH [5] or RSA-PSS [7], give signatures of length $|m| + R_\kappa$. For signature schemes based on RSA one should therefore consider schemes that provide some kind of message recovery, such as the message recovering variant of RSA-PSS, if bandwidth is a limiting factor. Indeed, there may even be situations where bandwidth is so precious that it is necessary to use elliptic-curve based schemes with message recovery, such as [15].

We consider the similar situation for public key encryption schemes. The standard elliptic curve encryption scheme for arbitrary-length messages is ECIES, which is a hybrid scheme modelled in the KEM+DEM framework [10]. This produces ciphertexts of length $E_\kappa + |m| + \kappa$. For RSA the hybrid scheme of choice is RSA-KEM [16], which leads to ciphertexts of length $R_\kappa + |m| + \kappa$. While the hybrid constructions are suitable for long messages, sufficiently short messages may be encrypted using only a single group element using a purely asymmetric encryption scheme, such as RSA-OAEP, [6, 17]. With RSA-OAEP, a message of maximal length $|m|$ is encoded together with two strings, each of length that of a hash function output, as a single group element of size $R_\kappa = |m| + 4\kappa + 2$ bits.

Our initial question was whether the bandwidth requirements for hybrid encryption schemes in the KEM+DEM framework may be reduced, in particular that of the RSA-based schemes. In the usual KEM+DEM framework, the KEM is used to encrypt a symmetric key, while the DEM is used to encrypt the message itself (using a symmetric encryption scheme and the key from the KEM). However, existing KEMs typically encode the symmetric key as a group element, and hence require either R_κ or E_κ bits.

This disparity in expansion rate grows when the security level increases, as the size of the groups grows much faster than the size of symmetric components. This is not so much a problem for elliptic curve systems where the growth is linear, but for factoring based systems the growth is quite pronounced, as shown in Fig. 1.

For "long" messages, the overhead inherent in the key encapsulation is quite negligible. However, in constrained environments it may be of significance, particularly when the messages being sent are on (roughly) the same order of magnitude as the size of the group element representation. This motivates the design and analysis of protocols that focus on keeping the message expansion to a minimum, at the expense of some additional algorithmic complexity.

Since R_κ in particular grows much faster than κ, it is natural to consider whether we may embed part of the message as part of the key encapsulation, and

Symmetric key length (κ)	Size of RSA modulus (R_κ)	Size of elliptic curve (E_κ)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

Fig. 1. Suggested parameter sizes (in bits) to maintain comparable security levels between different primitives, as recommended by NIST [14].

recover it during the decapsulation process. Such an approach was suggested for the specific case of RSA-OAEP in [16], but to the best of the authors' knowledge no formal analysis of that composition has been published, nor is it mentioned in the final ISO/IEC standard for public-key encryption [17].

In this paper we present a general concept of KEMs with message recovery, so-called RKEMs. We define security models for RKEMs, and prove a composition theorem which shows that a secure RKEM may be combined with an IND-CCA and INT-CCA secure DEM to obtain an IND-CCA secure public key encryption scheme. We then present a concrete example of an RKEM based on the RSA-OAEP construction, which is secure in the random oracle model. In combination with a standard Encrypt-then-MAC DEM, this results in a public key encryption scheme with ciphertexts of length as low as $|m|+5\kappa+3$ bits, i.e. a scheme whose messages does not depend on the size of the RSA modulus while providing κ-bit security. We then extend the concept of RKEMs to the Tag-KEM framework proposed by Abe et.al. [1] and propose a tag-RKEM based on the RSA-OAEP construction, which is secure in the random oracle model and will encrypt a message of length $|m|$ as a ciphertext of length $|m| + 4\kappa + 2$. This is the most space-efficient RSA-based construction known for long messages.

2 Definitions

2.1 Public Key Encryption

Our goal is to create bandwidth-efficient public key encryption schemes for arbitrary-length messages.

Definition 1 (Public-Key Encryption Scheme). *We define a public-key encryption scheme* $PKE = (PKE.Gen, PKE.Enc, PKE.Dec)$ *as an ordered tuple of three algorithms:*

1. *A probabilistic key generation algorithm* $PKE.Gen$*. It takes as input a security parameter* 1^κ*, and outputs a private/public keypair* (sk, pk)*. As part of the public key there is a parameter* $PKE.msglen$ *that specifies the maximum length of messages that can be encrypted in a single invocation; this value may be infinite.*

2. *A probabilistic encryption algorithm PKE.Enc. It takes as input a public key pk and a message m of length at most PKE.msglen, and outputs a ciphertext c.*
3. *A deterministic decryption algorithm PKE.Dec. It takes as input a private key sk and a ciphertext c, and outputs either a message m or the unique error symbol $\perp$.*

An encryption scheme must be sound, in the sense that the identity $m = PKE.Dec\big(sk, PKE.Enc(pk, m)\big)$ holds for valid keypairs $(sk, pk) = PKE.Gen(1^\kappa)$. Although we assume throughout this paper that our cryptographic primitives are perfectly sound, extension of our results to the case of non-perfect soundness is straightforward, with standard techniques.

To discuss the security of an encryption scheme, we must define a formal notion of what it means to "break" the scheme. Encryption schemes are designed with the goal of *confidentiality* in mind: an adversary should not be able to learn any useful information about encrypted messages. An encryption scheme is said to be IND-secure if there does not exist an efficient adversary who, given an encryption of two equal length messages, can determine which messages was encrypted to form the ciphertext.

In practice, we want to construct public-key encryption schemes that maintain confidentiality even under adaptive chosen-ciphertext attack (-CCA), which means that the adversary is allowed adaptive access to a decryption oracle. To quantify the concept of security, we compare the success of an adversary with the trivial "attack" of flipping a coin and guessing at random, and require that the (asymptotic) gain must be a *negligible* function in the security parameter 1^κ.

Definition 2 (Negligible function). *A function $f : \mathbb{N} \to \mathbb{R}$ is said to be* negligible *if for every polynomial p in $\mathbb{N}[x]$, there exists an $x_0 \in \mathbb{N}$ such that $f(n) \leq \frac{1}{|p(x)|}$ for all $x > x_0$.*

We now define the IND-CCA attack game for public-key encryption.

Definition 3 (IND-CCA Game for PKE). *The IND-CCA game for a given public-key encryption scheme PKE is played between the challenger and a two-stage adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, which is a pair of probabilistic Turing machines. For a specified security parameter 1^κ, the game proceeds as follows:*

1. *The challenger generates a private/public keypair $(sk, pk) = PKE.Gen(1^\kappa)$.*
2. *The adversary runs $\mathcal{A}_1$ on the input pk. During its execution, $\mathcal{A}_1$ may query a decryption oracle $\mathcal{O}_D$, that takes a ciphertext c as input, and outputs $PKE.Dec(sk, c)$. The algorithm terminates by outputting state information s and two messages m_0 and m_1 of equal length.*
3. *The challenger picks a bit $b \xleftarrow{R} \{0, 1\}$ uniformly at random, and computes $c^* = PKE.Enc(pk, m_b)$.*
4. *The adversary runs $\mathcal{A}_2$ on the input (c^*, s). During its execution, $\mathcal{A}_2$ has access to the decryption oracle as before, with the limitation that it may not ask for the decryption of the challenge ciphertext c^*. The algorithm terminates by outputting a guess b' for the value of b.*

We say that $\mathcal{A}$ wins *the IND-CCA game whenever* $b = b'$. *The* advantage *of* $\mathcal{A}$ *is the probability*

$$\mathrm{Adv}_{\mathrm{PKE}}^{\mathrm{IND-CCA}}(\mathcal{A}) = \left| Pr[\mathcal{A} \textit{ wins}] - 1/2 \right|.$$

A scheme is said to be IND-CCA secure if the advantage of every polynomial-time adversary is negligible (as a function of the security parameter). For specific schemes, we rarely have unconditional security, and speak instead of security with respect to some well-known reference problem that is thought to be difficult (such as the RSA or Diffie-Hellman problems).

2.2 Hybrid Encryption

Hybrid encryption is the practice of constructing public-key encryption schemes using a symmetric-key cipher as a building block. Although this adds a certain amount of complexity, it also enables PKE schemes to handle messages of more or less unrestricted length, at a low computational cost. The standard construction paradigm for hybrid encryption schemes is the KEM+DEM framework [10], in which a scheme is divided into two parts: an asymmetric *Key Encapsulation Mechanism* (KEM) and a symmetric *Data Encapsulation Mechanism* (DEM). Not only does this allow the encryption of arbitrary length messages but it also means that the PKE scheme obtained, by generically combining a secure KEM and a secure DEM, is itself secure [10]. This means that the components can be analysed separately and combined in a mix-and-match fashion.

Various modifications and variations of the basic KEM+DEM construction have also been proposed. In particular, we note the Tag-KEM schemes proposed in [1]. In Appendix A we present the standard definitions of a KEM and a DEM, how they are combined, and the appropriate security models for each.

3 KEMs with Partial Message Recovery

3.1 KEMs with Partial Message Recovery

In this section we introduce the notion of a KEM with partial message recovery (or RKEM). A KEM with partial message recovery is, quite simply, the use of public-key encryption to transmit a symmetric key and some partial message data in a secure manner. An obvious instantiation of such a scheme is to use a secure non-hybrid PKE to encrypt the "payload", although alternate implementations may also be possible. We will show that the obvious construction is indeed secure, and that the resulting RKEM+DEM hybrid encryption produces shorter messages than the traditional KEM+DEM construction.

Definition 4 (RKEM). *We define a KEM with partial message recovery to be an ordered tuple* $RKEM = (RKEM.Gen, RKEM.Encap, RKEM.Decap)$ *consisting of the following algorithms:*

1. *A probabilistic key generation algorithm RKEM.Gen. It takes a security parameter* 1^κ *as input, and outputs a private/public keypair* (sk, pk)*. As part of the public key there are two parameters, RKEM.msglen and RKEM.keylen. The value RKEM.msglen, which we assume is finite, denotes the maximum amount of message data that may be stored in an encapsulation, and RKEM.keylen denotes the fixed length of the symmetric key generated by the RKEM.*
2. *A probabilistic key encapsulation algorithm RKEM.Encap. It takes as input a public key pk, and a message m of length at most RKEM.msglen. The algorithm terminates by outputting a symmetric key k of length RKEM.keylen and an encapsulation* ψ*.*
3. *A deterministic key decapsulation algorithm RKEM.Decap. It takes as input a private key sk and an encapsulation* ψ*. The algorithm outputs either the unique error symbol* $\perp$*, or a pair* (k, m) *consisting of a symmetric key k of RKEM.keylen bits, and a message m of RKEM.msglen bits.*

If *RKEM.Encap* and *RKEM.Decap* are run using a valid keypair (sk, pk) and the ψ output by the encapsulation algorithm is used as input to the decapsulation algorithm, then the probability of failure is assumed to be zero. Furthermore, the decapsulation algorithm is required to output the same values of k and m as were associated with the encapsulation algorithm.

3.2 Security Definitions for RKEMs

Since the plaintext message is given as input to the encapsulation algorithm, it is necessary to adopt two separate security requirements for such RKEMs. First, we define the IND-CCA game for an RKEM similarly to that for a regular KEM, in which the adversary tries to distinguish whether a given key is the one embedded in a specified encapsulation.

Definition 5 (IND-CCA for RKEM). *The IND-CCA game for a given* RKEM *is played between a challenger and an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*. For a particular security parameter* 1^κ*, the game runs as follows.*

1. *The challenger generates a keypair* $(sk, pk) = RKEM.Gen(1^\kappa)$*.*
2. *The adversary runs* $\mathcal{A}_1$ *on the input pk. During its execution,* $\mathcal{A}_1$ *may query a decapsulation oracle* $\mathcal{O}_D$ *that takes an encapsulation* ψ *as input, and outputs the result of computing* $RKEM.Decap(sk, \psi)$*. The algorithm terminates by outputting a message m of length at most RKEM.msglen bits, as well as some state information s.*
3. *The challenger computes* $(k_0, \psi^*) = RKEM.Encap(pk, m)$*, and draws another key* $k_1 \xleftarrow{R} \{0,1\}^{RKEM.keylen}$ *as well as a random bit* $b \xleftarrow{R} \{0,1\}$ *uniformly at random*
4. *The adversary runs* $\mathcal{A}_2$ *on the input* (s, k_b, ψ^*)*. During its execution,* $\mathcal{A}_2$ *has access to the decapsulation oracle as before, with the restriction that it may not ask for the decapsulation of the challenge* ψ^**. The algorithm terminates by outputting a guess* b' *for the value of b.*

We say that $\mathcal{A}$ wins the game whenever the guess was correct, i.e. $b = b'$. The advantage of $\mathcal{A}$ is given as

$$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{RKEM}}(\mathcal{A}) = \left|Pr[\mathcal{A}\ wins] - 1/2\right|.$$

The other criterion relates to the confidentiality of the message used as input, and is represented by adopting the notion of RoR-CCA security from [3, 13]. The term *RoR* stands for real-or-random, which is because in this security definition an adversary is unable to tell a valid encryption of a message, from a random ciphertext. It can be shown that RoR-CCA security is equivalent to indistinguishability with respect to the message, but we shall not apply this equivalence directly. Instead, we only require the RoR-CCA property of the RKEM to imply that the full hybrid encryption scheme is IND-CCA secure.

Definition 6 (RoR-CCA for RKEM). *The RoR-CCA game for KEMs with partial message recovery is defined as follows:*

1. *The challenger generates a keypair $(sk, pk) = RKEM.Gen(1^\kappa)$.*
2. *The adversary runs $\mathcal{A}_1$ on the input pk. During its execution, $\mathcal{A}_1$ may query a decapsulation oracle $\mathcal{O}_D$ that takes an encapsulation ψ as input, and outputs the result of computing $RKEM.Decap(sk, \psi)$. The algorithm terminates by outputting a message m_0 of length at most $RKEM.msglen$ bits, as well as some state information s.*
3. *The challenger generates a random message m_1, which is of the same length as m_0, a random bit $b \overset{R}{\leftarrow} \{0, 1\}$, and computes $(k^*, \psi^*) = RKEM.Encap(pk, m_b)$.*
4. *The adversary runs $\mathcal{A}_2$ on the input (s, k^*, ψ^*). During its execution, $\mathcal{A}_2$ has access to the decapsulation oracle as before, with the restriction that it may not ask for the decapsulation of the challenge ψ^*. The algorithm terminates by outputting a guess b' for the value of b.*

We say that $\mathcal{A}$ wins the game whenever the guess was correct, i.e. $b = b'$. In each case the advantage of $\mathcal{A}$ is

$$\mathrm{Adv}^{\mathrm{RoR-CCA}}_{\mathrm{RKEM}}(\mathcal{A}) = \left|Pr[\mathcal{A}\ wins] - 1/2\right|.$$

Note that IND-CCA security definition really is about the ability of the adversary to determine whether a specified key is real or random, and RoR-CCA security is about the ability of the adversary to determine whether the embedded message is real or random. Hence, a more accurate nomenclature would be K-RoR-CCA and M-RoR-CCA, but we use the above nomenclature to stress the link with prior security definitions for standard KEMs.

3.3 Security of the Composition of an IND-CCA and RoR-CCA secure RKEM and an IND-PA and INT-CTXT secure DEM

Combining an RKEM with a DEM is done in the straightforward manner:

Definition 7 (RKEM+DEM Construction). *Given an RKEM and a DEM where the keys output by the RKEM are of the correct length for use with the DEM, i.e. RKEM.keylen = DEM.keylen, we construct a hybrid PKE scheme as follows.*

- *The key generation algorithm PKE.Gen executes RKEM.Gen to produce a private / public keypair, and appends any necessary information about the operation of the DEM.*
- *The encryption algorithm PKE.Enc is implemented as follows.*
  1. *The message m is padded to be of size at least $RKEM.msglen - 1$.*
  2. *The message m is then split into two component $m^{(0)}$ and $m^{(1)}$, i.e. $m = m^{(0)}||m^{(1)}$, where $m^{(0)}$ is of length $RKEM.msglen - 1$.*
  3. *Set $v = 1$, unless $m^{(1)} = \emptyset$, in which case we set $v = 0$.*
  4. *Compute a key/encapsulation pair $(k, \psi) = RKEM.Encap(pk, m^{(0)}||v)$.*
  5. *If $v = 1$ then encrypt the remaining part of the message to obtain a ciphertext $\chi = DEM.Enc_K(m^{(1)})$, otherwise set $\chi = \emptyset$.*
  6. *Output the ciphertext $c = (\psi, \chi)$.*
- *The decryption algorithm PKE.Dec is implemented as follows.*
  1. *Parse the ciphertext to obtain $(\psi, \chi) = c$.*
  2. *Recover the key and message fragment from ψ by computing $(k, m^{(0)}||v) = RKEM.Decap(sk, \psi)$.*
  3. *If $k = \perp$, return $\perp$ and halt.*
  4. *If $v = 1$ and $\chi \neq \emptyset$, return $\perp$ and halt.*
  5. *If $v = 0$, return $m^{(0)}$ and halt.*
  6. *Compute $m^{(1)} = DEM.Dec_k(\chi)$.*
  7. *If $m^{(1)} = \perp$, return $\perp$ and halt.*
  8. *Output $m^{(0)}||m^{(1)}$.*

The soundness of the RKEM+DEM construction follows from the soundness of the individual RKEM and DEM.

In the case where $|m| \leq RKEM.msglen$, there are few practical reasons to use the hybrid setup at all, and this is included mainly to avoid placing any artificial restrictions on our allowable message space. Our definition is no longer optimal in this case, since there is no reason to encapsulate a symmetric key k at all. We note that an alternate definition could specify that $RKEM.Decap$ returns a binary string s instead of k and $m^{(1)}||v$, which may then be parsed and interpreted depending on the value of v. The distinction is not important for our analysis, and is omitted in the further discussion for the sake of clarity.

Theorem 1 (Security of RKEM+DEM). *If the underlying RKEM is both IND-CCA and RoR-CCA secure and the DEM is IND-PA and INT-CTXT secure*[4]*, then the above composition is IND-CCA secure.*

More precisely we have, that if there is an adversary $\mathcal{A}$ against the above public key scheme, then there are polynomial-time adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ and $\mathcal{B}_4$

[4] The security definitions for DEMs are given in the Appendix.

such that

$$\begin{aligned}\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{PKE}}(\mathcal{A}) \leq\ & 2 \cdot \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{RKEM}}(\mathcal{B}_1) + q_D \cdot \mathrm{Adv}^{\mathrm{INT-CTXT}}_{\mathrm{DEM}}(\mathcal{B}_2) \\ & + 2 \cdot \mathrm{Adv}^{\mathrm{RoR-CCA}}_{\mathrm{RKEM}}(\mathcal{B}_3) \ + \mathrm{Adv}^{\mathrm{IND-PA}}_{\mathrm{DEM}}(\mathcal{B}_4),\end{aligned}$$

where q_D is an upper bound on the number of decryption queries made by $\mathcal{A}$.

Proof. Let $\mathcal{A}$ denote our adversary against the hybrid PKE system and let Game 0 be the standard IND-CCA game for a PKE. We prove the security by successively modifying the game in which $\mathcal{A}$ operates. In Game i, we let T_i denote the event that $b = b'$. Hence

$$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{PKE}}(\mathcal{A}) = |\Pr[T_0] - 1/2|.$$

Let Game 1 be the same as Game 0 except that if the challenger is asked by the adversary to decrypt a ciphertext (ψ^*, χ), where ψ^* is equal to the encapsulation-part of the challenge, then it uses the key k^* output by the encapsulation function when it decrypts χ, i.e. it only uses the valid decryption algorithm associated to the RKEM to obtain $m^{(0)}$. Since we assume that our algorithms are perfectly sound this is purely, at this stage, a conceptual difference, i.e.

$$\Pr[T_0] = \Pr[T_1].$$

Game 2 proceeds identically to Game 1, except for in the computation of the second component χ^* of the challenge ciphertext, where a *random* key k' is used instead of the key k^* that was returned by *RKEM.Encap*. It is clear that there exists a machine $\mathcal{B}_1$, whose running time is essentially that of $\mathcal{A}$, which can turn a distinguisher between the two games into an adversary against the IND-CCA property of the RKEM. We have

$$|\Pr[T_1] - \Pr[T_2]| \leq 2 \cdot \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{RKEM}}(\mathcal{B}_1).$$

Let Game 3 be the same as Game 2 except that when the challenger is asked by the adversary to decrypt a ciphertext (ψ^*, χ), where ψ^* is equal to the encapsulation-part of the challenge, then it simply rejects the ciphertext. It is clear that there exists a machine $\mathcal{B}_2$, whose running time is essentially that of $\mathcal{A}$, which can turn a distinguisher between the two games into an adversary against the INT-CTXT property of the DEM. We have

$$|\Pr[T_2] - \Pr[T_3]| \leq q_D \cdot \mathrm{Adv}^{\mathrm{INT-CTXT}}_{\mathrm{DEM}}(\mathcal{B}_2).$$

In Game 4, we change the computation of the encapsulation so that it encapsulates a random string instead of the first part of the message $m^{(0)}$, but we encrypt the second part of the message as in Game 2. Again, it is clear that there exists a machine $\mathcal{B}_3$, whose running time is essentially that of $\mathcal{A}$, which can turn a distinguisher between the two games into an adversary against the RoR-CCA property of the RKEM. We have

$$|\Pr[T_3] - \Pr[T_4]| \leq 2 \cdot \mathrm{Adv}^{\mathrm{RoR-CCA}}_{\mathrm{RKEM}}(\mathcal{B}_3).$$

Finally, in Game 4 we note that the first component of the ciphertext is completely random and independent of any message, and that the second part of the ciphertext is an encryption under a completely random key k^*. Hence, the adversary in Game 4 is essentially just an algorithm $\mathcal{B}_4$ which is attacking the IND-PA property of the DEM, i.e.

$$|\Pr[T_4] - 1/2] \leq \mathrm{Adv}^{\mathrm{IND-PA}}_{\mathrm{DEM}}(\mathcal{B}_4).$$

Putting the above equalities together we obtain the stated result

$$\begin{aligned}
\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{PKE}}(\mathcal{A}) &= |\Pr[T_0] - 1/2| = |\Pr[T_1] - 1/2| \\
&= |(\Pr[T_1] - \Pr[T_2]) + (\Pr[T_2] - \Pr[T_3]) \\
&\quad +(\Pr[T_3] - \Pr[T_4]) + (\Pr[T_4] - 1/2)| \\
&\leq |\Pr[T_1] - \Pr[T_2]| + |\Pr[T_2] - \Pr[T_3]| \\
&\quad + |\Pr[T_3] - \Pr[T_4]| + |\Pr[T_4] - 1/2]| \\
&\leq 2\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{RKEM}}(\mathcal{B}_1) + q_D \cdot \mathrm{Adv}^{\mathrm{INT-CTXT}}_{\mathrm{DEM}}(\mathcal{B}_2) \\
&\quad + 2\mathrm{Adv}^{\mathrm{RoR-CCA}}_{\mathrm{RKEM}}(\mathcal{B}_3) + \mathrm{Adv}^{\mathrm{IND-PA}}_{\mathrm{DEM}}(\mathcal{B}_4).
\end{aligned}$$

□

3.4 Constructions of RKEMs

A secure RKEM may be instantiated from an IND-CCA secure PKE in the obvious manner: use the PKE to encrypt the state bit, the κ-bit symmetric session key, and $PKE.msglen - \kappa - 1$ bits of message payload. It is easy to show that this construction is both IND-CCA and RoR-CCA secure, and that this is tightly related to the IND-CCA security of the underlying PKE [5]. If we implement this trivial scheme using RSA-OAEP, the total length of ciphertexts will be $|m| + 6\kappa + 3$; in the particular case of $\kappa = 128$ we save roughly 2400 bits in comparison with RSA-KEM.

However, the efficiency of our construction can be improved even further for a particular class of IND-CCA secure public-key encryption schemes. Let $c = \mathcal{E}(pk, m; r)$ denote a public key encryption algorithm taking a random string r as auxiliary input, with associated decryption function $m = \mathcal{D}(sk, c)$. We say that a public key algorithm is *randomness recovering*, if the decryption algorithm $\mathcal{D}(sk, c)$ can be modified so that it returns not only m but also the randomness used to construct c, i.e. we have that if $c = \mathcal{E}(pk, m; r)$ then $(m, r) = \mathcal{D}(sk, c)$. Such a scheme is said to be secure if it is IND-CCA secure with respect to the message m, and is OW-CPA secure with respect to the pair (m, r).

[5] In particular, the adversary is being given a challenge for the RKEM consisting of the encapsulation ψ^*, and a decapsulation oracle that computes $(k, m^{(0)}||v) = RKEM.Decap(sk, \psi)$, where $k|m^{(0)}|v = PKE.Dec(sk, c)$. In both the IND-CCA and ROR-CCA games the goal of the adversary is to determine some property of part of the "plaintext", either k or $m^{(0)}$. Whenever PKE is itself IND-CCA this is clearly not feasible.

There exist various practical public key encryption schemes that are securely randomness recovering, including RSA-OAEP and any scheme constructed using the Fujisaki-Okamoto transform [12]. In both of these constructions the IND-CCA security is standard, whilst the OW-CPA security with respect to the pair (m, r) follows from the OW-CPA security of the underlying primitive. We can use this to create a RKEM which incurs less overhead compared to the maximal message length of the underlying PKE.

Our IND-CCA and RoR-CCA secure RKEM is constructed as follows:

- *RKEM.Gen* is defined to be the key generation of the public key scheme, plus the specification of a hash function H. The parameter *RKEM.msglen* is a single bit less than the maximum message length of the public key scheme, and *RKEM.keylen* is the output length of H.
- *RKEM.Encap* takes a message m of length *RKEM.msglen*. It then computes $\psi = \mathcal{E}(pk, m; r)$ for some randomness r, $k = H(m||r)$ and returns (k, ψ).
- *RKEM.Decap* takes the encapsulation ψ and decrypts it using $\mathcal{D}(sk, \psi)$. If $\mathcal{D}(sk, \psi)$ returns $\perp$, then the decapsulation algorithm returns $\perp$ and halts. Otherwise the pair (m, r) is obtained and the algorithm proceeds to compute $k = H(m||r)$ and returns (m, k).

The RoR-CCA security of the above RKEM construction follows from the IND-CCA security of the underlying public key encryption scheme. The IND-CCA security follows, in the random oracle model, from the OW-CPA security of the underlying primitive with respect to the pair (m, r), using essentially the same proof of security as for standard RSA-KEM [16].

As mentioned in the introduction, by using RSA-OAEP in this construction one can obtain a public key encryption algorithm for messages of length m, which outputs ciphertexts of length $|m| + 5\kappa + 3$ bits for a given security parameter κ. This breaks down as follows: the RSA-OAEP encryption scheme (as defined in the ISO/IEC standard for public-key encryption [17]) has overhead of $2\,Hash.len + 2$ bits, where *Hash.len* is the length of the output from a cryptographic hash function, commonly taken to be 2κ bits[6]. Furthermore, a single state bit is used to keep track of the message length inside the RKEM. Finally, the usual method of constructing a DEM that is INT-CTXT requires a κ-bit message authentication code (MAC). In comparison with RSA-KEM using $\kappa = 128$, this scheme saves more than 2500 bits per message!

As we see, the above construction gives ciphertexts that are independent of the size of the RSA modulus used, being linear in the security parameter. Furthermore, we are able to extend the limited message space of the underlying RSA-based primitive "optimally", with only $\kappa + 1$ bits of overhead!

4 Tag-KEMs with Partial Ciphertext Recovery

After having discussed KEMs with partial message recovery it is natural to look at other formal models that exist for hybrid encryption schemes. In this section

[6] Although it may appear from the original OAEP paper that this should be only κ bits, it is necessary to use 2κ bits to deal with a more realistic attack model [2].

we consider Tag-KEMs [1], which have recently come into prominence as an attractive alternative to traditional KEMs. The main difference from regular KEMs is that the key encapsulation is used to preserve the integrity of the ciphertext, in addition to confidentiality of the symmetric key. The main result of [1] is that the use of Tag-KEMs makes it possible to create secure encryption schemes using a DEM that is only IND-PA. We define Tag-KEMs with partial ciphertext recovery (tag-RKEM) by direct extension of the previous definition in [1].

Definition 8 (Tag-KEM with Partial Ciphertext Recovery). *A Tag-KEM with partial ciphertext recovery (tag-RKEM) is defined as an ordered tuple of four algorithms.*

1. *A probabilistic algorithm $TKEM.Gen$ used to generate public keys. It takes a security parameter 1^κ as input, and outputs a private/public keypair (sk, pk). The public key includes all information needed for users of the scheme, including parameters specifying the length of the symmetric keys used ($TKEM.keylen$) and the size of the internal message space ($TKEM.msglen$).*
2. *A probabilistic algorithm $TKEM.Sym$ used to generate one-time symmetric keys. It takes a public key pk as input, and outputs a symmetric encryption key k and a string of internal state information s.*
3. *A probabilistic algorithm $TKEM.Encap$ used to encapsulate the symmetric key and part of the ciphertext. It takes some state information s as well as a tag τ as input, and outputs a key encapsulation ψ together with a suffix string $\tau^{(1)}$ of $\tau = \tau^{(0)}||\tau^{(1)}$ (consisting of the part of τ that may not be recovered from ψ).*
4. *A deterministic algorithm $TKEM.Decap$ used to recover the encapsulated key and ciphertext fragment from an encapsulation. It takes a private key sk, an encapsulation ψ and a partial tag $\tau^{(1)}$ as input, and outputs either a key k and the complete tag τ, or the unique error symbol $\perp$.*

A Tag-KEM is required to be sound in the obvious manner, i.e. for any τ and keypair (sk, pk) we have that $TKEM.Decap(sk, \psi, \tau^{(1)}) = (k, \tau)$ whenever $(\psi, \tau^{(1)}) = TKEM.Encap(\omega, \tau)$ and $(k, \omega) = TKEM.Sym(pk)$. We note that the definition collapses to that of [1] if we set $TKEM.msglen = 0$.

4.1 Security Definition for Tag-KEMs with Partial Ciphertext Recovery

A Tag-RKEM is said to be IND-CCA secure if there does not exist an adversary who can distinguish whether a given key k^* is the one embedded in an encapsulation ψ^*. The adversary has access to a decapsulation oracle, and is allowed to choose the tag τ^* used in the challenge encapsulation adaptively, but may not query the decapsulation oracle on the corresponding $(\psi^*, \tau^{(1)*})$. This corresponds to the notion of IND-CCA security for RKEMs used in the previous section, and is directly analogous to the security definition for regular Tag-KEMs [1].

Definition 9 (IND-CCA Security of Tag-RKEM). *For a given security parameter 1^κ, the IND-CCA game played between the challenger and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ runs as follows.*

1. *The challenger generates a private / public keypair $(sk, pk) = TKEM.Gen(1^\kappa)$.*
2. *The adversary runs $\mathcal{A}_1$ on the input pk. During its execution, $\mathcal{A}_1$ may query a decapsulation oracle $\mathcal{O}_{Decap}(\cdot,\cdot)$ which takes an encapsulation ψ and a tag $\tau^{(1)}$ as input, and returns the result of computing $TKEM.Decap(sk, \psi, \tau^{(1)})$. The algorithm terminates by outputting some state information s_1.*
3. *The challenger computes $(k_0, \omega) = TKEM.Sym(pk)$, and samples another key $k_1 \xleftarrow{R} \{0,1\}^{TKEM.keylen}$ uniformly at random. He then selects a random bit $b \xleftarrow{R} \{0,1\}$.*
4. *The adversary runs $\mathcal{A}_2$ on the input (s_1, k_b). During its execution, $\mathcal{A}_2$ has access to the same oracle as before. The algorithm terminates by outputting some state information s_2 and a tag τ^*.*
5. *The challenger generates a challenge encapsulation*

$$(\psi^*, \tau^{(1)*}) = TKEM.Encap(s_1, \tau^*).$$

6. *The adversary runs $\mathcal{A}_3$ on the input $(s_2, \psi^*, \tau^{(1)*})$. During its execution, $\mathcal{A}_3$ has access to the same oracle as before, with the restriction that the challenge $(\psi^*, \tau^{(1)*})$ may not be queried. The algorithm terminates by outputting a guess b' of the value of b.*

We say that $\mathcal{A}$ wins the game whenever the guess was correct, i.e. $b = b'$. The advantage of $\mathcal{A}$ is given as

$$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{TKEM}}(\mathcal{A}) = \left|Pr[\mathcal{A}\ wins] - 1/2\right|.$$

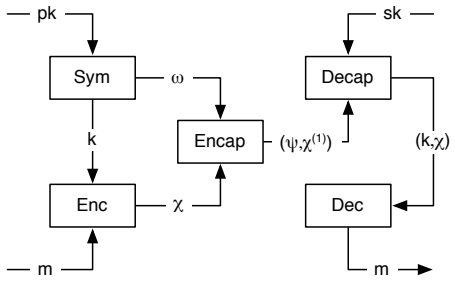

Fig. 2. Data flow in the Tag-KEM + DEM construction.

The security definition for Tag-RKEMs is versatile, in the sense that for a Tag-RKEM to be IND-CCA it must not only ensure that its symmetric keys are indistinguishable from random with respect to their encapsulations, but also

enforce certain non-malleability conditions with respect to τ. In particular, since the adversary is able to submit decapsulation oracle queries adaptively on ψ and $\tau^{(1)}$, the decapsulation procedure must be non-malleable in the sense that oracle queries such as $(\psi^*, \tau^{(1)})$ or $(\psi, \tau^{(1)*})$ reveal no information about k^*.

4.2 Security of the Composition of an IND-CCA secure Tag-RKEM and an IND-PA secure DEM

Combining a Tag-RKEM with a DEM is done by using the key from *TKEM.Sym* with *DEM.Enc* to produce a ciphertext, and using the resulting ciphertext as the tag for *TKEM.Encap*. The overall data-flow is illustrated in Fig. 2.

Definition 10 (TKEM+DEM Construction). *Given a TKEM and a DEM where the keys output by the TKEM are of correct length for use with the DEM, we construct a hybrid PKE scheme as follows.*

- *The key generation algorithm PKE.Gen is implemented by using RKEM.Gen and appending any necessary information about the DEM to the public key.*
- *The encryption algorithm PKE.Enc is implemented as follows.*
  1. *Compute a symmetric key* $(k, \omega) = TKEM.Sym(pk)$.
  2. *Compute the symmetric ciphertext* $\chi = DEM.Enc_k(m)$.
  3. *Create a key encapsulation using the ciphertext* χ *as the tag, by computing* $(\psi, \chi^{(1)}) = TKEM.Encap(\omega, \chi)$.
  4. *Output the ciphertext* $c = (\psi, \chi^{(1)})$.
- *The decryption algorithm PKE.Dec is implemented as follows.*
  1. *Parse the ciphertext to obtain* $(\psi, \chi^{(1)}) = c$.
  2. *Recover* k *and* χ *from* ψ *and* $\chi^{(1)}$ *by computing* $(k, \chi) = TKEM.Decap(sk, \psi, \chi^{(1)})$.
  3. *If TKEM.Decap returned* $\perp$*, return* $\perp$ *and halt.*
  4. *Recover the original message by running* $m = DEM.Dec_k(\chi)$.
  5. *If DEM.Dec returned* $\perp$*, return* $\perp$ *and halt.*
  6. *Output* m.

The soundness of the above construction follows from the soundness of the individual tag-RKEM and DEM. We note that this construction embeds part of the symmetric *ciphertext* rather than plaintext in the encapsulation, which explains why we no longer require RoR-CCA security (with respect to the message). This fact simplifies security analysis a great deal.

Theorem 2 (Security of TKEM+DEM). *If the underlying Tag-KEM with partial ciphertext recovery is IND-CCA secure and the DEM is IND-PA secure, then the TKEM+DEM composition is secure.*

More precisely we have, that if there is an adversary $\mathcal{A}$ *against the above hybrid public key scheme, then there are polynomial-time adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *such that*

$$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{PKE}}(\mathcal{A}) \leq 2 \cdot \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{TKEM}}(\mathcal{B}_1) + \mathrm{Adv}^{\mathrm{IND-PA}}_{\mathrm{DEM}}(\mathcal{B}_2).$$

Proof. Let $\mathcal{A}$ denote our adversary against the hybrid PKE system, and let Game 0 be the standard IND-CCA game for a PKE. We prove the security by making a single modification to Game 0, which causes the maximal advantage of any $\mathcal{A}$ to be clearly bounded. We define T_0 and T_1 to be the event that $b = b'$ in Games 0 and 1.

Let Game 1 be the same as Game 0, except that the challenger creates the symmetric ciphertext using a key k^* picked uniformly at random, rather than the key output by $TKEM.Sym$. It is clear that there exists a machine $\mathcal{B}_1$ whose running time is essentially that of $\mathcal{A}$, which can turn a distinguisher between the two games into an adversary against the IND-CCA property of the TKEM. Hence we have that

$$|\Pr[T_0] - \Pr[T_1]| = 2 \cdot \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{TKEM}}(\mathcal{B}_1).$$

However, in Game 1 the value of the challenge encapsulation ψ^* reveals *no* information about the key k^* used to create the ciphertext χ, since the symmetric key k^* was sampled independently of ω^*. Hence, the only information about the value of m_b available to $\mathcal{A}$ is the symmetric ciphertext fragment $\chi^{(1)*}$ (and plausibly the full symmetric ciphertext χ^*, depending on how the TKEM is constructed). Furthermore, the adversary is not able to make any adaptive decryption queries under k^*. Hence there exists a machine $\mathcal{B}_2$ whose running time is essentially that of $\mathcal{A}$, such that

$$|\Pr[T_1] - 1/2| = \mathrm{Adv}^{\mathrm{IND-PA}}_{\mathrm{DEM}}(\mathcal{B}_2).$$

Summarising, we obtain the stated result.

$$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{PKE}}(\mathcal{A}) \leq 2 \cdot \mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{TKEM}}(\mathcal{B}_1) + \mathrm{Adv}^{\mathrm{IND-PA}}_{\mathrm{DEM}}(\mathcal{B}_2).$$

□

4.3 Constructions of Tag-KEMs with partial ciphertext recovery

Having established that Tag-KEMs with partial ciphertext recovery are viable in principle, it remains to suggest a practical instantiation. We generalise a construction of Dent [11] to the tag-RKEM setting. This constructions makes use of an IND-CPA encryption scheme $(PKE.Gen, PKE.Enc, PKE.Dec)$ and two hash functions H and KDF. We suppose that the message space of the encryption scheme is $PKE.msglen$ and we use the notation $PKE.Enc(m, pk; r)$ to denote applying the encryption algorithm to a message m with a public key pk using random coins r. We require two security properties of the encryption scheme: that it is γ-uniform and that it is partially one-way.

Definition 11 (γ-uniform). *An encryption scheme*

$$(PKE.Gen, PKE.Enc, PKE.Dec)$$

is γ-uniform if, for all possible messages m and ciphertexts c

$$Pr[PKE.Enc(m, pk) = c] \leq \gamma$$

where the probability is taken over the random coins of the encryption algorithm.

Definition 12 (POW-CPA). *An encryption scheme (PKE.Gen, PKE.Enc, PKE.Dec) is partially one way with respect to inputs of length $\ell \leq PKE.msglen$ if no probabilistic polynomial-time attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ can win the following game with non-negligible probability:*

1. *The challenger generates a key pair $(pk, sk) = PKE.Gen(1^\kappa)$.*
2. *The attacker runs $\mathcal{A}_1$ on the input pk. $\mathcal{A}_1$ terminates by outputting a tag τ of length $PKE.msglen - \ell$ and some state information s,*
3. *The challenger randomly chooses a seed value α of length ℓ, sets $m = \alpha||\tau$ and computes $c^* = PKE.Enc(m, pk)$.*
4. *The attacker runs $\mathcal{A}_2$ on the input (c^*, s). $\mathcal{A}_2$ terminates by outputting a guess α' for α.*

The attacker wins if $\alpha' = \alpha$.

Note that if the encryption scheme is IND-CPA and ℓ is super-poly-logarithmic as a function of the security parameter, then the encryption scheme is POW-CPA secure.

We assume that we can split *PKE.msglen* into two lengths *TKEM.keylen* and *TKEM.msglen* such that $PKE.msglen = TKEM.keylen + TKEM.msglen$. We construct the tag-RKEM as follows:

1. *TKEM.Gen* runs *PKE.Gen* to generate a public/private key pair, and appends the values of *TKEM.keylen* and *TKEM.msglen* to the public key.
2. *TKEM.Sym* picks a random seed α of length *TKEM.keylen* and derives a key $k = KDF(\alpha)$. The algorithm outputs the state information $s = (pk, \alpha)$ and the key k.
3. *TKEM.Encap* runs in several steps:
   (a) Parse s as (pk, α).
   (b) Parse τ as $\tau^{(0)}||\tau^{(1)}$ where $\tau^{(0)}$ is *TKEM.msglen*-bits long if τ contains more than *TKEM.msglen* bits and $\tau^{(0)} = \tau$ if τ is less than *TKEM.msglen* bits in length.
   (c) Compute $m = \alpha||\tau^{(0)}$.
   (d) Compute $r = H(\alpha, \tau)$.
   (e) Compute $\psi = PKE.Enc(m, pk; r)$.
   (f) Output ψ.
4. $TKEM.Decap$ runs in several steps:
   (a) Compute $m = PKE.Dec(\psi, sk)$.
   (b) Parse m as $\alpha||\tau^{(0)}$ where α is *TKEM.keylen* bits in length.
   (c) If $\tau^{(0)}$ is less than *TKEM.msglen* bits in length and $\tau^{(1)} \neq \emptyset$, then output $\perp$.
   (d) Compute $\tau = \tau^{(0)}||\tau^{(1)}$.
   (e) Compute $r = H(\alpha||\tau)$.
   (f) If $\psi \neq PKE.Enc(m, pk; r)$ then output $\perp$.
   (g) Otherwise output $k = KDF(\alpha)$.

Theorem 3. *Suppose there exists an attacker $\mathcal{A}$ against the tag-RKEM in the random oracle model that makes at most q_D decapsulation oracle queries, q_K queries to the KDF-oracle, q_H queries to the H-oracle, and breaks the IND-CCA security with advantage $Adv^{\mathrm{IND-CCA}}_{\mathrm{TKEM}}$. Then there exists an attacker $\mathcal{B}$ against the POW-CPA security of the public key encryption scheme (with respect to the length TKEM.keylen) with advantage*

$$Adv^{\mathrm{POW-CPA}}_{\mathrm{PKE}} \geq \frac{1}{q_D + q_K + q_H}\left\{Adv^{IND-CCA}_{TKEM} - q_D/2^{TKEM.keylen} - q_D\gamma\right\}$$

Corollary 1. *If TKEM.keylen grows super-poly-logarithmically, γ is negligible and $(PKE.Gen, PKE.Enc, PKE.Dec)$ is partially one-way with respect to the length TKEM.keylen, then the tag-KEM construction is secure.*

If we instantiate this construction using the RSA-OAEP encryption scheme and a passively secure DEM, then the result construction will encrypt a message of length n using $n + 4\kappa + 2$ bits. This saves $\kappa + 1$ more bits than the RKEM construction given in Section 3.

5 Acknowledgements

This work was partially supported by the European Commission through the IST Programme under Contract IST-2002-507932 ECRYPT.

The problem of studying the efficiency and provable security of partial-message-recovery systems was suggested to the authors by Daniel J. Bernstein. Bernstein, who had incorporated a partial-message-recovery system (similar to that of Shoup [16]) into the benchmarking tools for the eBATS project [8].

The authors would also like to thank Mihir Bellare for explaining some subtle points related to the RSA-OAEP scheme and to James Birkett for reviewing a draft of the paper.

References

1. A. Abe, R. Gennaro, K. Kurosawa, and V. Shoup. Tag-KEM/DEM: A new framework for hybrid encryption and a new analysis of Kurosawa-Desmedt KEM. In *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 128–146, 2005.
2. M. Bellare. Personal correspondence, 2007.
3. M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. In *Proceedings of the 38th FOCS*, pages 394–403. IEEE, 1997.
4. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545, 2000.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

6. M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Advances in Cryptology - CRYPTO '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, 1995.
7. M. Bellare and P. Rogaway. The exact security of digital signatures — how to sign with RSA and Rabin. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, 1996.
8. D. J. Bernstein and T. Lange. eBATS (ECRYPT Benchmarking of Asymmetric Systems), 2007. `http://www.ecrypt.eu.org/ebats/`.
9. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532, 2001.
10. 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.
11. A. W. Dent. A designer's guide to KEMs. In *9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 133–151, 2003.
12. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Lecture Notes in Computer Science*, 1666:537–554, 1999.
13. D. Hofheinz, J. Mueller-Quade, and R. Steinwandt. On modeling IND-CCA security in cryptographic protocols. Cryptology ePrint Archive, Report 2003/024, 2003. `http://eprint.iacr.org/`.
14. National Institute of Standards and Technology. Recommendation for key management - part 1: General. Technical Report NIST Special Publication 800-57, National Institute of Standards and Technology, 2006.
15. L.A. Pintsov and S.A. Vanstone. Postal revenue collection in the digital age. In *Financial Cryptography 2001*, volume 1962 of *Lecture Notes in Computer Science*, pages 105–120, 2001.
16. V. Shoup. A proposal for an ISO standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. `http://eprint.iacr.org/2001/112/`.
17. V. Shoup. ISO/IEC FCD 18033-2 – Information technology – Security techniques – Encryption algorithms – Part 2: Asymmetric ciphers. Technical report, International Organization for Standardization, 2004. `http://shoup.net/iso/std6.pdf`.
18. A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications*, pages 324–328, Washington, DC, USA, 2005. IEEE Computer Society.

A KEM-DEM Framework

A.1 Definitions

Definition 13 (KEM). *A key encapsulation mechanism is defined by* $KEM = (KEM.Gen, KEM.Encap, KEM.Decap)$ *as an ordered tuple of three algorithms.*

1. *A probabilistic key generation algorithm* $KEM.Gen$*. It takes as input a security parameter* 1^κ*, and outputs a private/public keypair* (sk, pk)*. As part of the public key there is a parameter* $KEM.keylen$ *that specifies the length of the symmetric keys used by the DEM.*

2. *A probabilistic key encapsulation algorithm PKE.Encap. It takes as input a public key pk, and outputs a symmetric key k of length KEM.keylen, and an* encapsulation ψ.
3. *A deterministic decapsulation algorithm PKE.Decap. It takes as input a private key sk and an encapsulation ψ and outputs either a key k or the unique error symbol $\perp$.*

The KEM is sound if for almost all valid keypairs (sk, pk), whenever (k, ψ) was the output of $PKE.Encap(pk)$, we have $k = PKE.Decap(sk, \psi)$.

Definition 14 (DEM). *We define a data encapsulation mechanism $DEM = (DEM.Enc, DEM.Dec)$ as an ordered pair of algorithms.*

1. *A deterministic encryption algorithm DEM.Enc. It takes as input a message m and a symmetric key k of a specified length DEM.keylen, and outputs a ciphertext χ.*
2. *A deterministic decryption algorithm DEM.Dec. It takes as input a ciphertext χ and a symmetric key k of specified length, and outputs either a message m or the unique error symbol $\perp$.*

The DEM is sound as long as $m = DEM.Dec_k\big(DEM.Enc_k(m)\big)$ holds.

We assume that a DEM can take inputs of arbitrary length messages, thus the fact that the DEM can take a message of arbitrary length implies that any resulting hybrid encryption scheme can also take arbitrary length messages.

Definition 15 (KEM+DEM Construction). *Given a KEM and a DEM, where the keys output by the KEM are of correct length for use with the DEM, i.e. DEM.keylen = KEM.keylen, we construct a hybrid PKE scheme as follows.*

- *The key generation algorithm PKE.Gen is implemented using KEM.Gen.*
- *The encryption algorithm PKE.Enc is implemented as follows.*
  1. *Compute a key/encapsulation pair $(k, \psi) = KEM.Encap(pk)$.*
  2. *Encrypt the message to obtain a ciphertext $\chi = DEM.Enc_k(m)$.*
  3. *Output the ciphertext $c = (\psi, \chi)$.*
- *The decryption algorithm PKE.Dec is implemented as follows.*
  1. *Parse the ciphertext to obtain $(\psi, \chi) = c$.*
  2. *Compute the symmetric key $k = KEM.Decap(sk, \psi)$.*
  3. *If $k = \perp$, return $\perp$ and halt.*
  4. *Decrypt the message $m = DEM.Dec_k(\chi)$.*
  5. *If $m = \perp$, return $\perp$ and halt.*
  6. *Output m.*

The soundness of the KEM+DEM construction follows from the soundness of the individual KEM and DEM. A hybrid PKE scheme created from an IND-CCA secure KEM and an IND-CCA secure DEM is itself secure [10].

A.2 Security Models

For hybrid components such as KEMs and DEMs, we may adapt the indistinguishability criterion for public key schemes for each. For a KEM, the fundamental requirement is to ensure that the adversary does not learn anything about a key from its encapsulation. Indeed, given a key and an encapsulation the adversary should not be able to tell whether a given key is the one contained in an encapsulation. This is (slightly confusingly, since it is really a question of whether the key presented is "real" or "random") usually referred to as IND-CCA security.

Definition 16 (IND-CCA Game for KEM). *The IND-CCA game for a given key encapsulation mechanism KEM is played between the challenger and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. For a specified security parameter 1^κ, the game proceeds as follows.*

1. *The challenger generates a private/public keypair $(sk, pk) = KEM.Gen(1^\kappa)$.*
2. *The adversary runs $\mathcal{A}_1$ on the input pk. During its execution, $\mathcal{A}_1$ may query a decapsulation oracle $\mathcal{O}_D$ that takes an encapsulation ψ as input, and outputs $KEM.Decap(sk, \psi)$. The algorithm terminates by outputting some state information s.*
3. *The challenger generates a real key and its encapsulation, by calling $(k_0, \psi^*) = KEM.Encap(pk)$, as well as a random key k_1 drawn uniformly from the keyspace of the KEM. It also picks a random bit $b \xleftarrow{R} \{0,1\}$.*
4. *The adversary runs $\mathcal{A}_2$ on the input (k_b, ψ^*, s). During its execution, $\mathcal{A}_2$ has access to the decapsulation oracle as before, but it may not ask for the decapsulation of ψ^*. The algorithm terminates by outputting a guess b' for the value of b.*

We say that $\mathcal{A}$ wins the IND-CCA game whenever $b = b'$. The advantage of $\mathcal{A}$ is the probability

$$\mathrm{Adv}^{\mathrm{IND-CCA}}_{\mathrm{KEM}}(\mathcal{A}) = \left| Pr[\mathcal{A} \text{ wins }] - 1/2 \right|.$$

For DEMs, we give two security notions, the first of which is an adaption of the above notion of IND-security with respect to the input message.

Definition 17 (IND-PA Game for DEM). *The IND-PA game for a given data encapsulation mechanism DEM is played between the challenger and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. For a specified security parameter 1^κ, the game proceeds as follows.*

1. *The challenger generates a random symmetric key k.*
2. *The adversary runs $\mathcal{A}_1$ on the input 1^κ. The algorithm terminates by outputting two messages m_0 and m_1 of equal length, and some state information s.*
3. *The challenger generates a random bit $b \xleftarrow{R} \{0,1\}$ and encrypts the plaintext m_b, by calling $\chi^* = DEM.Enc_k(m_b)$.*

4. *The adversary runs $\mathcal{A}_2$ on the input (χ^*, s). The algorithm terminates by outputting a guess b' for the value of b.*

We say that $\mathcal{A}$ wins the IND-PA game whenever $b = b'$. The advantage of $\mathcal{A}$ is the probability

$$\mathrm{Adv}_{\mathrm{DEM}}^{\mathrm{IND-PA}}(\mathcal{A}) = \left| Pr[\mathcal{A} \text{ wins}] - 1/2 \right|.$$

Note, that a stronger notion of security exists which is IND-CCA, in this notion we give the adversary in both stages access to a decryption oracle with respect to the challengers key, subject to the constraint that in the second stage the adversary may not call the decryption oracle on the challenge ciphertext.

The other security notion we require is ciphertext integrity or INT-CTXT. This was first defined in [4], however we will only require a one-time security notion.

Definition 18 (INT-CTXT Game for DEM). *The INT-CTXT game for a given data encapsulation mechanism DEM is played between the challenger and an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$. For a specified security parameter 1^κ, the game proceeds as follows.*

1. *The challenger generates a random symmetric key k.*
2. *The adversary runs $\mathcal{A}_1$ on the input 1^κ. During its execution, $\mathcal{A}_1$ may query a decryption oracle $\mathcal{O}_D$ with respect to the key k; that takes a ciphertext χ as input, and outputs $DEM.Dec_k(\chi)$. The algorithm terminates by outputting a single messages m, and some state information s.*
3. *The challenger encrypts the plaintext m, by calling $\chi^* = DEM.Enc_k(m)$.*
4. *The adversary runs $\mathcal{A}_2$ on the input (χ^*, s). As before the adversary has access to it decryption oracle $\mathcal{O}_D$, however it may not call its oracle on the target ciphertext χ^*. The algorithm terminates by outputting a ciphertext $\chi' \neq \chi^*$.*

The adversary wins the game if it the ciphertext χ' is a valid ciphertext, i.e. it can be decrypted by the decryption algorithm. The advantage of $\mathcal{A}$ is the probability

$$\mathrm{Adv}_{\mathrm{DEM}}^{\mathrm{INT-CTXT}}(\mathcal{A}) = Pr[\mathcal{A} \text{ wins}].$$

It is proved in [4] that in the many-time setting, a scheme which is both IND-PA and INT-CTXT will be IND-CCA as well. It is straightforward to verify that this property also holds for one-time encryption schemes.

We note that a symmetric cipher which is IND-PA (such as a secure block cipher in CBC mode with fixed IV) can be made INT-CTXT by adding a secure Message Authentication Code using the basic Encrypt-then-MAC construction. This is also the "standard" way of producing an IND-CCA symmetric cipher [10].

B Proof of Theorem 3

In this section we prove the the tag-RKEM construction given in Section 4.3 is secure. In other words, we prove the following theorem:

Theorem 3. *Suppose there exists an attacker $\mathcal{A}$ against the tag-RKEM in the random oracle model that makes at most q_D decapsulation oracle queries, q_K queries to the KDF-oracle, q_H queries to the H-oracle, and breaks the IND-CCA security with advantage $Adv^{\mathrm{IND-CCA}}_{\mathrm{TKEM}}$. Then there exists an attacker $\mathcal{B}$ against the POW-CPA security of the public key encryption scheme (with respect to the length TKEM.keylen) with advantage*

$$Adv^{\mathrm{POW-CPA}}_{\mathrm{PKE}} \geq \frac{1}{q_D + q_K + q_H}\left\{Adv^{\mathrm{IND-CCA}}_{TKEM} - q_D/2^{TKEM.keylen} - q_D\gamma\right\}$$

Proof. The proof is very similar to the construction of Dent [11]. We model both the hash function H and the key derivation function KDF as random oracles. The tag-RKEM attacker can then gain no advantage in determining whether the challenge key k^* is correct key or not unless the attacker queries the KDF oracle on the challenge seed value α^*. This can be done either implicitly (by making a valid decapsulation oracle query that uses the value α^* as its seed) or explicitly by querying the KDF-oracle directly. We show that it is computationally infeasible for the attacker to make a valid decapsulation oracle query using the seed α^* without querying the H-oracle on some message $\alpha^*||\tau$. Hence, the only way that the attacker can gain a non-negligible advantage is to query one of the random oracles with a value involving α^*. We can therefore recover α^*, and solve the POW-CPA problem by guessing which oracle query contains α^*.

We use non-programmable random oracles. These random oracles are simulated using two lists KDFList and HList. In both cases, when a query is made to the random oracle on an input x, then oracle searches the relevant list for a record (x, y). If such a record exists, then the oracle outputs y; otherwise, the oracle generates a random value y of the appropriate size, adds (x, y) to the appropriate list, and outputs y.

Again, we use a game-hopping proof. Let T_i be the event that the tag-RKEM attacker wins in Game i. Let Game 0 be the normal IND-CCA attack game for $\mathcal{A}$. Hence,

$$Adv^{\mathrm{IND-CCA}}_{\mathrm{TKEM}} = |Pr[T_0] - 1/2| \ .$$

Let Game 1 be identical to Game 0 except that the attacker is immediately deemed to have lost the game if, on conclusion of the game, it turns out that $\mathcal{A}$ queried the decapsulation oracle on the challenge ciphertext ψ^* *before* the challenge ciphertext was issued. (We are forced to do this as the simulated decapsulation oracle that we will define would incorrectly decapsulate this ciphertext as $\perp$). Let E_1 be the event that $\mathcal{A}_1$ submits ψ^* to the decapsulation oracle. Since $\mathcal{A}_1$ has no information about ψ^* at this point, this would require $\mathcal{A}_1$ to guess the value of ψ^*, which implicitly means that $\mathcal{A}_1$ has guessed the value of α^* as ψ^* uniquely defines the value of α^*. Hence, $Pr[E_1] \leq q_D/2^{TKEM.keylen}$ and we obtain the following relation:

$$|Pr[T_0] - Pr[T_1]| \leq Pr[E_1] \leq q_D/2^{TKEM.keylen} \ .$$

Game 2 will be identical to Game 1 except that we change the decapsulation oracle to the following simulated decapsulation oracle:

1. The oracle takes as input an encapsulation ψ and a tag τ. It parses τ as $\tau^{(0)}||\tau^{(1)}$.
2. The oracle searches HLIST for an entry with input values (α, τ) and output value r such that $\psi = PKE.Enc(\alpha||\tau^{(0)}, pk; r)$.
3. If such a record exists, then the oracle outputs $KDF(r)$; otherwise, it output $\perp$.

Game 2 functions identically to Game 1 unless the attacker makes a decapsulation oracle query which is valid in Game 1 but declared invalid in Game 2. Let E_2 be the event that this occurs. The only way that E_2 can occur is if $\mathcal{A}$ submits a ciphertext ψ and tag τ for decapsulation such that $H(\alpha, \tau) = r$ and $\psi = PKE.Enc(\alpha||\tau^{(0)}, pk; r)$ but $\mathcal{A}$ has not submitted $\alpha||\tau$ to the H-oracle. This means that, in the view of the attacker, the value r is completely random. Hence, the probability that ψ is an encryption of $\alpha||\tau^{(0)}$ is bounded by γ and we obtain the following relation:

$$|Pr[T_1] - Pr[T_2]| \leq q_D \gamma \, .$$

We may now construct an attacker $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against the POW-CPA property of the encryption scheme and relate $\mathcal{B}$'s success probability to $\mathcal{A}$'s advantage in Game 2. $\mathcal{B}_1$ takes pk as input and runs as follows:

1. Run $\mathcal{A}_1$ on the input pk. Simulates the oracles to which $\mathcal{A}$ has access as in Game 2. $\mathcal{A}_1$ terminates by outputting some state information s_1.
2. Generates a random key k^*.
3. Run $\mathcal{A}_2$ on the input (k^*, s_1). Simulate the oracles to which $\mathcal{A}$ has access as in Game 2. $\mathcal{A}_2$ terminates by outputting some state information s_2 and a tag τ^*.
4. Parse τ^* as $\tau^{*(0)}||\tau^{*(1)}$.
5. Output the tag $\tau^{*(0)}$.

The challenger will now pick a random seed α^* and form the challenge ciphertext ψ^* by encrypting $\alpha^*||\tau^{*(0)}$. $\mathcal{B}_2$ takes ψ^* as input and runs as follows:

1. If $\mathcal{A}_1$ or $\mathcal{A}_2$ made a decapsulation oracle query on the value ψ^*, then output $\perp$ and halt.
2. Run A_3 on the input ψ^* and s_2. Simulate the oracles to which $\mathcal{A}$ has access as in Game 2. $\mathcal{A}_3$ terminates by outputting a bit b'.
3. Randomly choose an entry from the total number of records in KDFLIST and HLIST, and extract the value α from this query. Output α as the guess for α^*.

$\mathcal{B}$ completely simulates the oracles to which $\mathcal{A}$ has access up until the point that $\mathcal{A}$ makes a KDF-oracle query on α^* or an H-oracle query on $\alpha^*||\tau$. If $\mathcal{A}$ does not make such a query, then its advantage is 0; hence, the probability that $\mathcal{A}$ makes such a query is equal to $\mathcal{A}$'s advantage in Game 2.

Therefore, the probability that $\mathcal{B}$ correctly solves the POW-CPA problem is equal to the probability that $\mathcal{A}$ makes a KDF-oracle query on α^* or an H-oracle query on $\alpha^*||\tau$, and $\mathcal{B}$ correctly guesses a record that contains a reference to α^*.

Since there are at most $q_D + q_K + q_H$ records in KDFLIST and HLIST, this will happen with probability at least $1/q_D + q_K + q_H$. Hence,

$$\begin{aligned} Adv_{PKE}^{\mathrm{POW-CPA}} &\geq \frac{1}{q_D + q_K + q_H} \cdot |Pr[T_2] - 1/2| \\ &\geq \frac{1}{q_D + q_K + q_H} \Big\{ Adv_{TKEM}^{\mathrm{IND-CCA}} - q_D/2^{TKEM.keylen} - q_D\gamma \Big\} \end{aligned}$$

This proves the theorem. □