SK-KEM : AN IDENTITY-BASED KEM

M. BARBOSA, L. CHEN, Z. CHENG, M. CHIMLEY, A. DENT, P. FARSHIM, K. HARRISON, J. MALONE-LEE, N.P. SMART, AND F. VERCAUTEREN

1. Description of the cryptographic technique

1.1. **Design Rationale.** In this document we will describe an identity-based encryption scheme based on the work of Sakai and Kasahara [19], and which we call SK-KEM. The scheme operates on any elliptic curve group which has an efficiently computable bilinear pairing and does not require the use of super-singular curves. The resulting scheme is the most efficient and flexible provably-secure scheme we have seen in the literature.

Our scheme is intended to be instantiated on an elliptic curve group with an efficiently computable Weil or Tate pairing. However, without loss of generality, we present the scheme as acting over an arbitrary group structure with an efficiently computable bilinear map. Hence, we consider a situation where:

- $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are (multiplicative) cyclic groups of prime order p.
- g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$.
- $\hat{e}$ is a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$.

The map $\hat{e}$ must have the following properties.

- It must be bilinear, i.e. for all $u \in \mathbb{G}_1$, all $v \in \mathbb{G}_2$ and all $a, b \in \mathbb{Z}$ we have $\hat{e}(u^a, v^b) = \hat{e}(u, v)^{ab}$.
- It must be non-degenerate, i.e. $\hat{e}(g_1, g_2) \neq 1$.
- It must be efficiently computable, i.e. there must exist an efficient algorithm to compute $\hat{e}(u, v)$ for all $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$. In addition the group laws in the three groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ must be efficiently computable.

It should be noted that while we present the scheme as acting over arbitrary groups, many of the performance advantages of the scheme are obtained by considering practical methods of instantiating these groups as elliptic curve groups. We are able to make significant performance gains by avoiding certain (computationally expensive) group operations.

In designing the SK-KEM scheme, we considered the following design criteria, in decreasing order of importance:

(1) We wished to design a scheme that followed the KEM-DEM design philosophy. This philosophy is incredibly flexible and has many advantages (as discussed in Section 2.1). In particular, it allows the asymmetric and symmetric parts of the cryptosystem to be chosen independently and allows for the security level of the scheme to be chosen independently of the maximum message size.

(2) We wished to design a scheme that was as efficient as possible (for all practical instantiations of the bilinear group definition). In particular this means:
   - We avoid hashing into $\mathbb{G}_1$ and $\mathbb{G}_2$. This avoids cofactor exponentiations in the groups in which $\mathbb{G}_1$ and $\mathbb{G}_2$ are embedded.
   - Wherever possible, we avoid group exponentiation in $\mathbb{G}_2$, as this group is typically embedded in a group that is much larger than $\mathbb{G}_1$ and group exponentiation is correspondingly more expensive. In particular, we avoid all group exponentiations in $\mathbb{G}_2$ in the encapsulation and decapsulation algorithms.
   - Wherever possible, we reduce bandwidth requirements by transmitting elements of $\mathbb{G}_1$ as opposed to $\mathbb{G}_2$.
   - Wherever possible, we allow the precomputation of pairing values. Our scheme requires the online computation of only one pairing value (as part of the decapsulation algorithm).
   - We avoid using an isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ in the scheme, although it is used in the security proof. See [20] for a discussion of the use of the isomorphism ψ within proofs.

   The efficiency of the scheme is considered in Section 2.2.
(3) We wished to design a scheme that was provably secure and whose security could be reduced to some recognised hard problem. However, because we considered the practicality of the scheme to be more important than any theoretical security arguments, we were prepared to accept proofs of security that are presented in the random oracle model. The security considerations are discussed in Section 3.
(4) Since many uses of identity-based encryption involve encrypting the same message to multiple recipients, we wished to design a scheme that can be easily extended to provide efficient encryption to multiple parties. Multi-party encryption is discussed in Section 1.5.
(5) We wished to design a scheme that was patent free. We are unaware of any patents that affect the SK-KEM scheme — see Section 5.

The remainder of this section will describe the general construction of an asymmetric encryption scheme from a KEM and a DEM, the Sakai-Kasahara KEM, the general Sakai-Kasahara primitive and, lastly, will discuss methods for constructing suitable DEMs. We will attempt to present these schemes in a format consistent with the current IEEE 1363 editorial policy, so as to easily facilitate the inclusion of the algorithm in the proposed standard.

1.2. **KEM-DEM constructions.** The KEM-DEM construction method for an asymmetric encryption scheme involves the combination of an asymmetric *key encapsulation mechanism* (KEM) and a symmetric *data encapsulation mechanism* (DEM), and is based on the work of Cramer and Shoup [8]. The KEM-DEM construction is a highly flexible and efficient construction paradigm (see Section 2.1).

In general, an identity-based encryption scheme consists of four polynomial-time algorithms:

- $\mathbb{G}_{\mathtt{ID}}(1^t)$: A probabilistic, polynomial-time *master key generation* algorithm which takes as input 1^t and returns a master public key $M_{\mathfrak{pk}}$ and a master secret key $M_{\mathfrak{sk}}$.
- $\mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{sk}}, \mathtt{ID})$: A (possibly probabilistic), polynomial-time *private key extraction* algorithm which takes as input $M_{\mathfrak{sk}}$ and $\mathtt{ID} \in \{0,1\}^*$, an identifier string, and returns the associated private key $D_{\mathtt{ID}}$.
- $\mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathfrak{pk}}, m; r)$: A probabilistic, polynomial-time *encryption* algorithm. On input of an identifier $\mathtt{ID}$, the master public key $M_{\mathfrak{pk}}$, a message $m \in \mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}})$ and possibly some randomness $r \in \mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}})$ this algorithm outputs $c \in \mathbb{C}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.
- $\mathbb{D}_{\mathtt{ID}}(D_{\mathtt{ID}}, c)$: A deterministic, polynomial-time *decryption* algorithm. On input of the private key $D_{\mathtt{ID}}$ and a ciphertext c this outputs the corresponding value of the plaintext m or a failure symbol $\perp$.

We now demonstrate how to construct an identity-based encryption scheme from an identity-based KEM and a DEM.

1.2.1. *Scheme Options.* The following options shall be established or otherwise agreed upon between the parties to the scheme (the sender and the recipient):

- An identity-based KEM scheme, which should be SK-KEM (Section 1.3) or an identity-based KEM scheme described in an addendum to this standard.
- A DEM scheme, which should be selected from those presented in Section 1.6 or a DEM scheme described in an addendum to this standard.
- The symmetric key length, which should be the equal to the length of the symmetric keys output by the KEM and to the length of the symmetric key taken as input by the DEM.
- Any scheme options required by the KEM scheme or DEM scheme.

The above information may remain the same for any number of executions of the identity-based encryption scheme, or it may be changed at some frequency. The information need not be kept secret.

1.2.2. *Master Key Generation.* The master key generation algorithm of the identity-based encryption scheme $\mathbb{G}_{\mathtt{ID}}$ shall be equal to the master key generation algorithm of the identity-based KEM scheme $\mathbb{G}_{\mathtt{ID-KEM}}$.

1.2.3. *Private Key Extraction.* The private key extraction algorithm of the identity-based encryption scheme $\mathbb{X}_{\mathtt{ID}}$ shall be equal to the private key generation algorithm of the identity-based KEM scheme $\mathbb{X}_{\mathtt{ID-KEM}}$.

1.2.4. *Encryption Operation.* The ciphertext (C_1, C_2) shall be generated by a sender from a message m and the recipient's identity $\mathtt{ID}$ by the following or an equivalent sequence of steps:

(1) Compute a KEM encapsulation (K, C_1) from the recipient's identity $\mathtt{ID}$ with the KEM encapsulation algorithm.
(2) Compute a DEM encryption C_2 from the message m and the symmetric key K with the DEM encryption algorithm.
(3) Output (C_1, C_2).

1.2.5. *Decryption Operation.* The plaintext m shall be recovered from the ciphertext (C_1, C_2) by the following or an equivalent sequence of steps:

(1) Select the private key $D_{\mathtt{ID}}$ for the operation.
(2) Compute the KEM decapsulation K from the ciphertext C_1 and the selected private key $D_{\mathtt{ID}}$ with the KEM decapsulation algorithm. If the algorithm returns "error", output "error" and stop.
(3) Compute the plaintext m from the ciphertext C_2 and the symmetric key K with the DEM decryption algorithm. If this algorithm returns "error", output "error" and stop.
(4) Output m.

1.3. **The Sakai-Kasahara KEM (SK-KEM).** In this section we present the Sakai-Kasahara identity-based KEM (SK-KEM). An identity-based KEM consists of four polynomial-time algorithms:

- $\mathbb{G}_{\mathtt{ID-KEM}}(1^t)$: The PPT *master key generation algorithm* which takes as input 1^t. It outputs a master public key $M_{\mathfrak{pk}}$ and a master secret key $M_{\mathfrak{sk}}$.
- $\mathbb{X}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, M_{\mathfrak{sk}}, \mathtt{ID})$: The PPT *private key extraction algorithm* which takes as input $M_{\mathfrak{pk}}, M_{\mathfrak{sk}}$ and $\mathtt{ID} \in \{0,1\}^*$, an identifier string. It outputs the associated private key $D_{\mathtt{ID}}$.
- $\mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID})$: The PPT *encapsulation algorithm* which takes as input $\mathtt{ID}$ and $M_{\mathfrak{pk}}$. It outputs a pair (k, c) where $k \in \mathbb{K}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}})$ is a key and $c \in \mathbb{C}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}})$ is the encapsulation of that key.
- $\mathbb{D}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID}, D_{\mathtt{ID}}, c)$: The deterministic *decapsulation algorithm* which takes as input $M_{\mathfrak{pk}}$, $\mathtt{ID}$, c and $D_{\mathtt{ID}}$. It outputs k or a failure symbol $\perp$.

The SK-KEM scheme is based on the Sakai-Kasahara method of constructing identity-based private keys [19] and makes use of the generic ID-KEM construction of Bentahar *et al.* [2] — which is itself based on one of the generic KEM constructions of Dent [9]. The SK-KEM scheme was originally proposed by Chen *et al.* [6] (who also present a full security proof).

An informal presentation of the scheme is given in Table 1. Note that the scheme makes use of four cryptographic hash functions

$$\begin{aligned} H_1 &: \{0,1\}^* \longrightarrow \mathbb{Z}_p, \\ H_2 &: \mathbb{G}_T \longrightarrow \{0,1\}^n, \\ H_3 &: \{0,1\}^n \longrightarrow \mathbb{Z}_p, \\ H_4 &: \{0,1\}^n \longrightarrow \{0,1\}^{\ell_d}, \end{aligned}$$

where ℓ_d is the bit length of the keys required by the DEM with which the KEM will be used and n is a value derived from the security parameter. Suggested values for n are given in Section 3. In practice, we expect that each hash function H_i (with possible exception of H_4) will be implemented as

$$H_i(x) = H(I2OSP(i,1)||x)$$

where H is a particular standardised hash function chosen depending on the security parameter (see Section 3). Hash function H_4 may be implemented in this way, or it may be implemented as the IEEE 1363 key derivation function KDF1 – this allows for the easy selection of an arbitrary output length.

The remainder of this section will be devoted to a more formal description of the SK-KEM scheme.

TABLE 1. The SK-KEM

$\mathbb{G}_{\mathtt{ID}}(1^t)$: • $s \leftarrow \mathbb{Z}_p^*$ • $R \leftarrow g_1^s$ The value s is the master secret key $M_{\mathfrak{sk}}$ of the TA, while R along with the other system parameters is the master public key $M_{\mathfrak{pk}}$.	$\mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{pk}}, \mathtt{ID}, s)$: This outputs the identity-based secret key $$D_{\mathtt{ID}} = g_2^{1/(s+H_1(\mathtt{ID}))}.$$ Note this will fail and, moreover, $M_{\mathfrak{sk}}$ will be revealed if $H_1(\mathtt{ID}) = -s$; however, this happens with negligible probability.
$\mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID})$ • $m \leftarrow \{0,1\}^n$ • $r \leftarrow H_3(m)$ • $Q \leftarrow R \cdot g_1^{H_1(\mathtt{ID})}$ • $U \leftarrow Q^r$ • $V \leftarrow m \oplus H_2(\hat{e}(g_1, g_2)^r)$ • $k \leftarrow H_4(m)$ • $c \leftarrow (U, V)$ • Return (k, c)	$\mathbb{D}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID}, D_{\mathtt{ID}}, c)$ • Parse c as (U, V) • $\alpha \leftarrow \hat{e}(U, D_{\mathtt{ID}})$ • $m \leftarrow H_2(\alpha) \oplus V$ • $r \leftarrow H_3(m)$ • $Q \leftarrow R \cdot g_1^{H_1(\mathtt{ID})}$ • If $U \neq Q^r$, return $\perp$ • $k \leftarrow H_4(m)$ • Return k

1.3.1. *Scheme Options.* The following options shall be established or otherwise agreed upon between the parties to the scheme (the sender and the recipient):

- the length n used by the encapsulation algorithm
- the length ℓ_d of symmetric keys that the SK-KEM is to produce

The above information may remain the same for any number of executions of the encryption scheme, or it may be changed at some frequency. If the above information is changed, then a new master key shall be generated and all recipients shall be issued with new private keys. The above information need not be kept secret.

1.3.2. *Master Key Generation.* The SK-KEM master key generation algorithm shall be an approved algorithm for computing a Sakai-Kasahara master pair (see Section 1.4.1).

1.3.3. *Private Key Extraction.* The Sakai-Kasahara private key for an identity ID shall be computed by the following or by an equivalent sequence of steps:

(1) Obtain the recipient's purported identity ID.
(2) (Optional) Validate the identity ID. Stop if validation fails.
(3) Compute the SK private key $D_{\mathtt{ID}}$ from ID and the master private key s with the primitive SK-XP (Section 1.4.2).

1.3.4. *Encapsulation Operation.* A symmetric key K and the Sakai-Kasahara encapsulation (U, V) shall be generated by a sender from the recipient's identity by the following or an equivalent sequence of steps:

(1) Obtain the master public key R.
(2) (Optional) Validate the master public key R. Stop if validation fails.
(3) Randomly choose a bitstring m of length n.
(4) Compute the value $r = H_3(m)$

(5) Compute a secret shared string z and an encapsulation value U from the identity $\mathtt{ID}$, the master public key R and the integer r with the SK-SSVD primitive (Section 1.4.3).
(6) Compute $V = z \oplus m$
(7) Compute $K = H_4(m)$.
(8) Output the symmetric key K and the encapsulation (U, V).

1.3.5. *Decapsulation Operation.* A symmetric key K shall be recovered from the encapsulation (U, V) and the recipient's Sakai-Kasahara private key $D_{\mathtt{ID}}$ by the following or an equivalent sequence of steps:

(1) Obtain the master public key R.
(2) (Optional) Validate the master public key R. Stop if validation fails.
(3) Compute the secret shared string z from the encapsulation value U and the SK private key $D_{\mathtt{ID}}$ with the SK-RSVD primitive (Section 1.4.4).
(4) Compute $m = z \oplus V$.
(5) Compute $r = H_3(m)$.
(6) Compute $Q = R \cdot g_1^{H_1(\mathtt{ID})}$
(7) If U does not equal Q^r, then output "error" and stop.
(8) Compute $K = H_4(m)$.
(9) Output the symmetric key K.

1.4. **The Sakai-Kasahara Primitive.** In this section we describe the Sakai-Kasahara encryption and decryption primitives. These algorithms work with Sakai-Kasahara master keys and Sakai-Kasahara private keys.

1.4.1. *Sakai-Kasahara Master Keys.* A Sakai-Kasahara master key pair consists of

- An SK *master public key*, which is an integer $s \in \mathbb{Z}_p^*$,
- An SK *master private key*, which is a group element $R = g_1^s \in \mathbb{G}_1$.

We suggest that SK master key pairs are computed as follows:

(1) Randomly select an integer s from $\mathbb{Z}_p^*$.
(2) Compute $R = g_1^s$.
(3) Output (s, R)

Optionally, $\hat{e}(g_1, g_2)$ can be included in an SK master public key to minimise the number of pairing computations necessary in later computations.

An SK master key pair should only be generated by a trusted, privileged party.

1.4.2. *Sakai-Kasahara Private Keys (SK-XP).* The Sakai-Kasahara private key for an identity $\mathtt{ID}$ consists of a group element $D_{\mathtt{ID}} \in \mathbb{G}_2$. The SK private key $D_{\mathtt{ID}}$ for an identity $\mathtt{ID}$ shall be computed by the following or an equivalent sequence of steps:

(1) Obtain the SK master key pair (s, R).
(2) (Optional) Validate the SK master key pair. Stop if validation fails.
(3) Compute the SK private key $D_{\mathtt{ID}} = g_2^{1/(s+H_1(\mathtt{ID}))}$.
(4) Output $D_{\mathtt{ID}}$.

An SK private key shall be computed by a trusted privileged party with knowledge of the SK master key pair.

1.4.3. *Sakai-Kasahara Sender's Secret Value Derivation Primitive (SK-SSVD).* SK-SSVD is the Sakai-Kasahara Encryption Primitive. It is invoked in the SK-KEM scheme as part of the encapsulation process, given the identity of the recipient, the master public key and an integer. The SK-RSVD primitive is used in the decapsulation process to generate the same shared secret value.

Input:

- The recipient's identity $\texttt{ID}$.
- The master public key R.
- An integer $r \in \mathbb{Z}_p$.

Output:

- An encapsulation value $U \in \mathbb{G}_1$.
- A shared secret value $z \in \{0,1\}^n$.

Operation: The output pair (U, z) shall be computed by the following or an equivalent sequence of steps:

(1) Let $Q = R \cdot g_1^{H_1(\texttt{ID})}$.
(2) Let $U = Q^r$.
(3) Let $z = H_2(\hat{e}(g_1, g_2)^r)$.
(4) Output (U, z).

1.4.4. *Sakai-Kasahara Receiver's Secret Value Derivation Primitive (SK-RSVD).* SK-RSVD is the Sakai-Kasahara Decryption Primitive. It is invoked in the SK-KEM scheme as part of the decapsulation process, given the recipient's SK private key and an encapsulation value. The SK-SSVD primitive is used in the encapsulation process to generate the same shared secret value.

Input:

- The recipient's SK private key $D_{\texttt{ID}}$.
- An encapsulation value $U \in \mathbb{G}_1$.

Output:

- A shared secret value $z \in \{0,1\}^n$.

Operation: The shared secret value z shall be computed by the following or an equivalent sequence of steps:

(1) Let $z = H_2(\hat{e}(U, D_{\texttt{ID}}))$.
(2) Output z.

1.5. **Extension to multiple recipients.** Any asymmetric encryption scheme is relatively inefficient when used to encrypt the same message for more than one recipient. Unless a scheme is specially adapted to handle multiple recipients (μ recipients, say), the sender has no choice but to encrypt the same message μ times. Such an operation will undoubtedly be unacceptably slow for even some relatively small values of μ and certainly requires the sender to have a large amount of bandwidth available.

If used naively, then a KEM-DEM scheme is no exception to this rule; however, the nature of the KEM-DEM construction lends itself to producing schemes that can be easily adapted for multiple recipients. If it is possible to arrange for a KEM to output the same key value for each recipient, then the sender only needs to

execute the DEM once and may result in faster KEM computations. It certainly reduces the bandwidth required for the ciphertext.

The SK-KEM scheme can be easily extended to the case of multiple recipient. An informative description of such a scheme is given in Table 2. This extension can be proven secure under a model of security for ID-KEMs to multiple recipients, such as that described in [1]. Whilst this extension is trivial, it can be proved secure

TABLE 2. Extension to multiple recipients

$\mathbb{E}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, \mathtt{ID}_1, \ldots, \mathtt{ID}_t)$	$\mathbb{D}_{\mathtt{ID-KEM}}(M_{\mathfrak{pk}}, j, \mathtt{ID}_j, D_{\mathtt{ID}_j}, c)$
• $m \leftarrow \{0,1\}^n$ • For $i = 1, \ldots, t$ – $r_i \leftarrow H_3(m \Vert \mathtt{ID}_i)$ – $Q_i \leftarrow R \cdot g_1^{H_1(\mathtt{ID}_i)}$ – $U_i \leftarrow Q_i^{r_i}$ – $V_i \leftarrow m \oplus H_2(\hat{e}(g_1, g_2)^{r_i})$ • $k \leftarrow H_4(m)$ • $c \leftarrow \{(U_i, V_i)\}_{i=1}^t$ • Return (k, c)	• Parse c as $\{(U_i, V_i)\}_{i=1}^t$ • $\alpha \leftarrow \hat{e}(U_j, D_{\mathtt{ID}_j})$ • $m \leftarrow H_2(\alpha) \oplus V_j$ • $r \leftarrow H_3(m \Vert \mathtt{ID}_j)$ • $Q_j \leftarrow R \cdot g_1^{H_1(\mathtt{ID}_j)}$ • If $U_j \neq Q_j^r$, return $\perp$ • $k \leftarrow H_4(m)$ • Return k

within an appropriate model [1]. However, the fact that the encryption operation avoids pairings means that this extension can be carried out more efficiently than for schemes which require a pairing in the encryption operation.

1.6. **Data Encapsulation Mechanisms.** A KEM-DEM encryption scheme is necessarily incompletely described unless a suitably secure DEM is given. A DEM is essentially a symmetric encryption algorithm that is secure against active attacks. While we do not intend to give a precise description of a particular DEM in this proposal, we would recommend that the 1363 committee adopt those DEMs standardised by the ISO/IEC joint technical committee [15]. Special consideration should also be given to standardising the MUTLI-S01 mode of operation of a stream cipher as a DEM [16] and the simple "Encrypt-then-MAC" scheme.

2. Claimed attributes and advantages of the technique

2.1. **The KEM-DEM Framework.** The KEM-DEM framework is a highly efficient and flexible paradigm for designing asymmetric encryption schemes. This framework was developed by Cramer and Shoup [8] and takes the well-known idea of using symmetric encryption within an asymmetric encryption scheme to its logical conclusion by completely separating out the asymmetric and symmetric parts of the cryptosystem. Bentahar *et al.* extended this idea to the identity-based setting [2].

For our purposes, the identity-based KEM-DEM framework splits an identity-based encryption scheme into two parts: an asymmetric identity-based *key encapsulation algorithm* (ID-KEM) and a symmetric *data encapsulation algorithm* (DEM). To encrypt a message, the user does two things:

- The user first runs the ID-KEM encapsulation algorithm using the recipient's identity as a public key. This algorithm outputs a random symmetric key of a pre-determined length, and an encapsulation (encryption) of that key.
- The user then encrypts the message using the DEM encryption algorithm and the randomly symmetric key produced by the ID-KEM.

The overall ciphertext consists of the encapsulated symmetric key output by the ID-KEM and the encrypted message output by the DEM. Such a ciphertext can be decrypted using the following two stage process:

- The user runs the ID-KEM decapsulation algorithm on the encapsulated key and the user's own private key. This algorithm terminates by outputting either a symmetric key or an error symbol. If the algorithm outputs an error symbol, then the user takes no further action.
- The user then decrypts the encrypted message using the DEM decryption algorithm and the symmetric key output by the ID-KEM. This algorithm outputs either the decrypted message or an error symbol.

The cryptosystems that result from following the KEM-DEM paradigm are typically simple, elegant and very efficient. The nature of symmetric and asymmetric cryptography is such that the efficiency costs of the symmetric DEM are typically inconsequential, and both the running time and the memory requirements for a KEM-DEM encryption scheme are dominated by the running time and memory requirements for the KEM. Hence, a KEM-DEM scheme is likely to be as efficient as any other asymmetric encryption scheme. There is a slight bandwidth overhead associated with using a KEM-DEM encryption scheme, as an n-bit message is encrypted to form a ciphertext that is more than n-bits long; however, the ciphertext expansion is only a constant number of bits and the resulting freedom to encrypt messages of any length is (in most cases) more than worth the slightly higher bandwidth costs.

The main advantage of a KEM-DEM encryption scheme is that it completely separates the size of the public key from the maximum size of the message that can be encrypted. Non-hybrid asymmetric encryption schemes (i.e. schemes that do not make use of symmetric techniques) are typically only applicable to messages whose length is bounded by a function of the public key. For example, in a non-hybrid RSA-based encryption scheme, such as RSA-OAEP [14], the message length is bounded by a fraction of the size of the modulus. This forces a user to either:

- choose a security level that is appropriate for the security requirements of their environment, and accept they can only encrypt short messages; or
- choose a security level that provides an encryption scheme that can encrypt messages of a suitable length for their application, and accept that the encryption scheme has a unnecessarily high security parameter and is comparatively inefficient.

Similar problems can occur in the identity-based setting. KEM-DEM constructions avoid these problems by decoupling the relationship between the size of the security parameter for the KEM and the message length. KEM-DEM constructions can be used to encrypt messages of any length and using an identity-based KEM that has its security parameter chosen solely based on the security requirements of the environment in which the application will be used.

The KEM-DEM framework is also highly adaptable. It has been proven [2] that any combination of secure ID-KEM and secure DEM provides a secure identity-based encryption scheme. Therefore, it is relatively easy to create applications with a large number of different choices for encryption, simply by providing a number of choices for the KEM and DEM. Furthermore, the framework admits many other types of functionality: by combining a DEM with a (normal) KEM or a signcryption KEM, the user can easily create a (normal) asymmetric encryption scheme or an outsider-secure signcryption scheme. This means that the user can easily select an encryption scheme that precisely matches their security requirements.

Another advantage of KEM-DEM encryption schemes is that they allow for efficiency gains through the use of pre-computation. Since the message is not required for the KEM encapsulation computation, the symmetric key and the encapsulation of that symmetric key can be pre-computed. This means that only the DEM encryption needs to be computed after the message is provided. The use of fast DEM encryption techniques, such as could be provided by a stream cipher using the MULTI-S01 combining function [16], could provide almost real-time asymmetric data encryption.

This approach to asymmetric encryption has already been adopted by other standard bodies. For example, in the ISO/IEC standard on asymmetric encryption [15], four of the six algorithms follow the KEM-DEM paradigm. Of course, if a body standardises a KEM, then they are also forced to standardise a DEM or they will not have achieved their aim of fully specifying an asymmetric encryption scheme. DEMs need to be able to resist active attacks against the confidentiality of the data that they are protecting. Fortunately, there are several simple and provably secure constructions for DEMs, including the simple "Encrypt-then-MAC" scheme.

2.2. **Efficiency Comparison.** We now compare our scheme to various other ID-based encryption schemes in the literature. We do not assume any optimizations involving folding together certain operations, i.e. we look at the operation counts of the schemes as presented in the various papers.

In [12] three types of pairing parameters are defined. The first of these corresponds to super-singular elliptic curves, and so suffers from scalability issues at high security levels. The second two refer to ordinary elliptic curves and are distinguished by the definition of the group $\mathbb{G}_2$. In Type 2 systems one has a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, but one cannot hash into $\mathbb{G}_2$. Type 3 are characterised by being able to hash into $\mathbb{G}_2$, but not being able to compute an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$. Type 3 systems can be implemented very efficiently using sextic twists, which gives improved performance for operations in the group $\mathbb{G}_2$, plus using curves with sextic twists enables the use of more efficient pairings, such as the Ate-pairing [11]. The main advantages of using Type 3 curves against Type 2 curves are

- Operations in $\mathbb{G}_2$ are much more efficient.
- Bandwidth is greatly reduced if transmitting/storing elements in $\mathbb{G}_2$.
- Hashing to elements of $\mathbb{G}_2$ is possible.

The only disadvantage in using Type 3 curves is that one does not have a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$.

We shall compare the protocols in terms of computational efficiency by reference to the cost of a multiplication in $\mathbb{G}_1$. Thus we need to know the relative efficiency of multiplication in $\mathbb{G}_2$ and exponentiation in $\mathbb{G}_T$, and the relative cost of pairings

to this base measure. The relative cost of multiplication in $\mathbb{G}_2$ and exponentiation in $\mathbb{G}_T$ is relatively easy to measure [12], but the relative cost of pairings is harder. Our assumption is that at the 128-bit security level we implement the systems using Type 3 [12] set of pairing parameters (i.e. asymmetric pairings with no computable isomorphism) using $k = 12$ and a sextic twist. We shall assume, following [13], that a pairing in this situation requires around 20 times the cost of a multiplication in $\mathbb{G}_1$. This implies we are assuming that a Tate pairing is used, as opposed to an Ate-pairing [11]. In Table 3 we summarise the estimates we will use in what follows.

TABLE 3. Relative cost of operations and bandwidth at the 128-bit security level, with $k = 12$. These are theoretical relative costs based on multiplication counts only, an actual implementation will have different relative costs.

	Type 3
Size of Elements	
$\in \mathbb{G}_1$	256 bits
$\in \mathbb{G}_2$	512 bits
$\in \mathbb{G}_T$	3072 bits
Relative Cost of Operation	
Mul in $\mathbb{G}_1$	1
Mul in $\mathbb{G}_2$	3
Exp in $\mathbb{G}_T$	3
Pairing	20
Hash in $\mathbb{G}_1$	free
Hash in $\mathbb{G}_2$	3

A detailed comparison of the exact operations required by each scheme is given in Table 4. Specific notes on each scheme are marked by superscripts in relation to the following list

(1) By BF we mean the Boneh-Franklin [4] scheme optimized for bandwidth whereby the group element transmitted in the ciphertext belongs to $\mathbb{G}_1$, by $\mathrm{BF}^{\perp}$ we mean the Boneh-Franklin scheme with the opposite form, namely when the group element transmitted in the ciphertext belongs to $\mathbb{G}_2$
(2) This refers to the second scheme in the paper by Boneh and Boyen from EuroCrypt 2004 [3].
(3) This refers to Water's scheme from EuroCrypt 2005 [21].
(4) This refers to the Kiltz KEM scheme [17]. We use the asymmetric version with the shortest ciphertexts.
(5) This refers to Gentry's scheme [10]. We use the CCA2 version from the proceedings based on the Cramer-Shoup model, a hybrid version was mentioned at the conference but has appeared in print yet. The scheme has a very tight security reduction, but to an even more non-standard problem than the q-BDHI problem.
(6) The scheme is proved secure in the selective ID security model, without random oracles. Extension to the general model is trivial, if one is to allow ROM based proofs. However, CCA2 security is obtained by appealing to a general NIZK construction. We do not count the cost of the additional measures for CCA2 security in the table.

(7) Again, CCA2 security is obtained by appealing to a general NIZK or HIBE construction. We do not count the cost of the additional measures for CCA2 security in the table.
(8) We count public key generation as a seperate operation from encryption and decryption, and so do not count the cost of public key generation as part of the encryption process.
(9) The construction of the public/private keys in the Water's scheme and the Kiltz scheme involves various group exponentiations depending on which bits are set in the identity string. This means the system parameters are rather large. For asymmetric pairings they are even larger, and the construction is rather involved. However, whilst the key construction method is involved and requires a large table of fixed group elements, it essentially costs the same as a group exponentiation in terms of time.
(10) This value represents a product of μ pairings which, it is claimed, can be computed more efficiently than μ independent pairings. However, the precise performance improvement depends on the type of pairing one is using and the security level. We therefore count each pairing as independent.

Combining the Table 3 and 4 together we can derive Table 5, which compares the relative costs of each primitive. Note, for the BB and Waters' schemes we do not count the cost of the CCA2 protection required.

This demonstrates that SK-KEM is the most efficient identity-based encryption scheme when instantiated at a 128-bit security level. At higher security levels, the SK-KEM scheme becomes progressively more efficient compared to the other schemes, both in terms of computational efficiency and ciphertext size.

2.3. **Informative Comparison.** We pass over briefly the advantages and disadvantages of the other schemes mentioned in the previous sections compared to our proposal.

Boneh-Franklin (BF) The Boneh-Franklin scheme has a proof of security in the random oracle model and bandwidth requirements which are similar to the Sakai-Kasahara KEM scheme. However, encryption using the Boneh-Franklin scheme is significantly more expensive than with the Sakai-Kasahara scheme, and it is necessary for every user to hash into $\mathbb{G}_2$ when computing another user's public key.

Transposed Boneh-Franklin ($\mathbf{BF}^{\perp}$) The transposed Boneh-Franklin scheme also has a security proof in the random oracle model; however, it's bandwidth requirements are slightly larger than the Saka-Kasahara scheme. Furthermore, these bandwidth requirement get worse as the security parameter increases. As with the "normal" Boneh-Franklin scheme, encryption is a very expensive operation.

Boneh-Boyen (BB) The Boneh-Boyen scheme has a proof of security in the standard model, which may be thought of as an advantage over the Sakai-Kasahara scheme. However, the Boneh-Boyen scheme discussed in the previous section is only IND-CPA secure. Any attempt to convert the Boneh-Boyen scheme into an IND-CCA2 secure encryption scheme either be very inefficient or will require the use of the random oracle model. Furthermore, the bandwidth requirements for the Boneh-Boyen scheme are large and get much worse as the security parameter increases.

Waters The Waters scheme has a similar property to the Boneh-Boyen scheme: it has a proof of security in the standard model, but this proof only proves that

TABLE 4. Comparison of schemes

	SK-KEM	BF[1]	BF$^{\perp}$ [1]	BB[2]	Waters[3]	Kiltz[4]	Gentry[5]
Properties							
KEM Design	Y	-	-	-	-	Y	-
ROM Proof	Y	Y	Y	-	-	-	-
Hard Problem	q-BDHI	BDH	BDH	q-BDHI[6]	D-BDH[7]	D-BDH	q-ABDHE
Private Key Generation							
Hash to $\mathbb{G}_1$	-	-	1	-	-	-	-
Hash to $\mathbb{G}_2$	-	1	-	-	-	-	-
Exp in $\mathbb{G}_1$	-	-	1	-	-	-	-
Exp in $\mathbb{G}_2$	1	1	-	1	2[9]	3[9]	6
Public Key Generation[8]							
Hash to $\mathbb{G}_1$	-	-	1	-	-	-	-
Hash to $\mathbb{G}_2$	-	1	-	-	-	-	-
Exp in $\mathbb{G}_1$	1	-	-	1	[9]	[9]	1
Exp in $\mathbb{G}_2$	-	-	-	-	-	-	-
Encrypt/ Encapsulate							
Exp in $\mathbb{G}_1$	1	2	1	3	2	2.5	1
Exp in $\mathbb{G}_2$	-	-	1	-	-	-	-
Exp in $\mathbb{G}_T$	1	-	-	-	1	1	3.5
Pairings	-	1	1	-	-	-	-
Decrypt/ Decapsulate							
Exp in $\mathbb{G}_1$	1	1	-	1	-	-	-
Exp in $\mathbb{G}_2$	-	-	1	-	-	2.5	1
Exp in $\mathbb{G}_T$	-	-	-	-	-	-	2
Pairings	1	1	1	1	2[10]	2[10]	2

the scheme is IND-CPA secure. Any attempt to convert the scheme into an IND-CCA2 scheme will either be very inefficient or will require the use of the random oracle model. The bandwidth requirements for the scheme are large and get much worse as the security parameter increases. The system parameters are large and, furthermore, decryption is a very slow operation in the Waters scheme.

Kiltz The Kiltz scheme does have a proof of security in the standard model, but its decryption algorithm is very slow and it has a very large public key.

Gentry The Gentry scheme does have a proof of security in the standard model. The proof is very tight which could mean, if one took tightness of security results to apply in practice which is debateable, that key sizes could be smaller. However, this tightness result does not take into account that parameter generation for asymmetric pairings is not simple. In addition the reduction is to a very non-standard

TABLE 5. Cost, relative to an exponentiation in $\mathbb{G}_1$ of the various schemes in Table 4

	SK-KEM	BF	$\text{BF}^{\perp}$	BB	Waters	Kiltz	Gentry
Ciphertext Size (Bits)	$256 + n$	$256 + 2n$	$512 + 2n$	3584	3584	512	9472
Private Key Generation	3	6	1	3	6	9	18
Public Key Generation	1	3	0	1	≈ 1	≈ 1	1
Encrypt/ Encapsulate	4	22	24	3	5	5.5	11.5
Decrypt/ Decapsulate	21	21	23	21	40	47.5	47

problem. Bandwidth is large, although one presumes this will be smaller for the hybrid version. Encryption and decryption are also slow compared to SK-KEM.

3. Security result and recommended key sizes

In the following two sections we fully describe the formal security model for identity-based KEMs, then we state the security result for our SK-KEM scheme and provide some recommended key sizes.

3.1. **Security Model.** In the following paragraphs we formally describe the security notion that we require for IBE as formalised by Boneh and Franklin [4]. We then proceed to give the security definitions that we require to build a secure IBE scheme using the KEM-DEM framework and state the composition theorem from [2].

3.1.1. *ID-Based Encryption Schemes.* Here we give the security notions for an ID-based encryption scheme, as first introduced by Boneh and Franklin [4].

We define the message, ciphertext and randomness spaces of scheme by $\mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}})$, $\mathbb{C}_{\mathtt{ID}}(M_{\mathfrak{pk}})$, $\mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}})$. These are parametrised by the master public key $M_{\mathfrak{pk}}$, and hence by the security parameter t. An ID-based encryption scheme is specified by four polynomial-time algorithms:

- $\mathbb{G}_{\mathtt{ID}}(1^t)$: A PPT algorithm which takes as input 1^t and returns a master public key $M_{\mathfrak{pk}}$ and a master secret key $M_{\mathfrak{sk}}$.
- $\mathbb{X}_{\mathtt{ID}}(M_{\mathfrak{sk}}, \mathtt{ID})$: A (possibly) probabilistic private key extraction algorithm which takes as input $M_{\mathfrak{sk}}$ and $\mathtt{ID} \in \{0,1\}^*$, an identifier string, and returns the associated private key $D_{\mathtt{ID}}$.
- $\mathbb{E}_{\mathtt{ID}}(\mathtt{ID}, M_{\mathfrak{pk}}, m; r)$: This is the PPT encryption algorithm. On input of an identifier $\mathtt{ID}$, the master public key $M_{\mathfrak{pk}}$, a message $m \in \mathbb{M}_{\mathtt{ID}}(M_{\mathfrak{pk}})$ and possibly some randomness $r \in \mathbb{R}_{\mathtt{ID}}(M_{\mathfrak{pk}})$ this algorithm outputs $c \in \mathbb{C}_{\mathtt{ID}}(M_{\mathfrak{pk}})$.
- $\mathbb{D}_{\mathtt{ID}}(D_{\mathtt{ID}}, c)$: This is the deterministic decryption algorithm. On input of the private key $D_{\mathtt{ID}}$ and a ciphertext c this outputs the corresponding value of the plaintext m or a failure symbol $\perp$.

Consider the following two-stage games between an adversary A of the encryption algorithm and a challenger.

ID-IND Adversarial Game

(1) $(M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID}}(1^t)$.
(2) $(s, \mathtt{ID}^*, m_0, m_1) \leftarrow A_1^{\mathcal{O}_{\mathtt{ID}}}(M_{\mathtt{pk}})$.
(3) $b \leftarrow \{0,1\}$.
(4) $c^* \leftarrow \mathbb{E}_{\mathtt{ID}}(\mathtt{ID}^*, M_{\mathtt{pk}}, m_b; r)$.
(5) $b' \leftarrow A_2^{\mathcal{O}}(c^*, s)$.

In the above, s is some state information and $\mathcal{O}$ are oracles to which the adversary has access. We describe these below.

The adversary has access to a private key extraction oracle and to a decryption oracle. It may call these oracles with identities of its choosing, subject to the restrictions that in the second phase A is not allowed to call the decryption oracle with the pair $(c^*, \mathtt{ID}^*)$ and that it never calls its private key extraction oracle with $\mathtt{ID}^*$. We define the adversary's advantage as

$$\mathrm{Adv}_{\mathtt{ID}}^{\mathtt{ID-IND-CCA}}(A) = |2\Pr[b' = b] - 1|.$$

An ID-based encryption algorithm is considered to be secure secure against *adaptive chosen ciphertext attack* if, for all PPT adversaries, the advantage in the relevant game is a negligible function of the security parameter t.

3.1.2. *ID-Based Key Encapsulation Mechanisms.* An ID-KEM scheme is specified by four polynomial-time algorithms:

- $\mathbb{G}_{\mathtt{ID-KEM}}(1^t)$. A PPT algorithm which takes as input 1^t and returns a master public key $M_{\mathtt{pk}}$ and a master secret key $M_{\mathtt{sk}}$.
- $\mathbb{X}_{\mathtt{ID-KEM}}(M_{\mathtt{sk}}, \mathtt{ID})$. A (possibly) probabilistic algorithm which takes as input $M_{\mathtt{sk}}$ and an identifier string, $\mathtt{ID} \in \{0,1\}^*$, and returns the associated private key $D_{\mathtt{ID}}$.
- $\mathbb{E}_{\mathtt{ID-KEM}}(\mathtt{ID}, M_{\mathtt{pk}})$. This is the PPT encapsulation algorithm. On input of $\mathtt{ID}$ and $M_{\mathtt{pk}}$ this outputs a pair (k, c) where $k \in \mathbb{K}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}})$ is a key and $c \in \mathbb{C}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}})$ is the encapsulation of that key.
- $\mathbb{D}_{\mathtt{ID-KEM}}(D_{\mathtt{ID}}, c)$. This is the deterministic decapsulation algorithm. On input of c and $D_{\mathtt{ID}}$ this outputs k or a failure symbol $\perp$.

Consider the following two-stage game between an adversary A of the ID-KEM and a challenger.

ID-IND Adversarial Game

(1) $(M_{\mathtt{pk}}, M_{\mathtt{sk}}) \leftarrow \mathbb{G}_{\mathtt{ID-KEM}}(1^t)$.
(2) $(s, \mathtt{ID}^*) \leftarrow A_1^{\mathcal{O}}(M_{\mathtt{pk}})$.
(3) $(k_0, c^*) \leftarrow \mathbb{E}_{\mathtt{ID-KEM}}(\mathtt{ID}^*, M_{\mathtt{pk}})$.
(4) $k_1 \leftarrow \mathbb{K}_{\mathtt{ID-KEM}}(M_{\mathtt{pk}})$.
(5) $b \leftarrow \{0,1\}$.
(6) $b' \leftarrow A_2^{\mathcal{O}}(c^*, s, k_b)$.

In the above s is some state information and $\mathcal{O}$ denotes oracles to which the adversary has access. We describe these below.

The adversary has access to a private key extraction oracle and to a decapsulation oracle. It may call these oracles with identities of its choosing, subject to the restrictions that in the second phase A is not allowed to call $\mathcal{O}_{\mathtt{ID}}$ with the pair $(c^*, \mathtt{ID}^*)$ and it never extracts the private key for $\mathtt{ID}^*$.

The adversary's advantage is defined to be

$$\mathrm{Adv}^{\mathtt{ID-IND-CCA}}_{\mathtt{ID-KEM}}(A) = |2\Pr[b' = b] - 1|.$$

An ID-KEM is considered to be secure, if its advantage is a negligible function of the security parameter t.

3.1.3. *One-Time Symmetric Encryption.* A one-time symmetric encryption scheme is a pair of deterministic polynomial time secret key (SK) algorithms, $\mathbb{E}_{\mathtt{SK}}$ and $\mathbb{D}_{\mathtt{SK}}$, where key, message and ciphertext spaces are given by $\mathbb{K}_{\mathtt{SK}}(t)$, $\mathbb{M}_{\mathtt{SK}}(t)$, $\mathbb{C}_{\mathtt{SK}}(t)$ for some security parameter t.

- $\mathbb{E}_{\mathtt{SK}}(k, m)$. On input of $k \in \mathbb{K}_{\mathtt{SK}}(t)$ and $m \in \mathbb{M}_{\mathtt{SK}}(t)$ this outputs a value $c \in \mathbb{C}_{\mathtt{SK}}(t)$.
- $\mathbb{D}_{\mathtt{SK}}(k, c)$. This performs the inverse operation, or outputs $\perp$ if c is not the encryption of a message m under the key k.

We will assume $\mathbb{M}_{\mathtt{SK}}(t) = \{0,1\}^*$ and that the scheme is sound: for all m we have $\mathbb{D}_{\mathtt{SK}}(k, \mathbb{E}_{\mathtt{SK}}(k, m)) = m$. We assume that the key length $|k|$ is a polynomial function of the security parameter t.

Security of one-time symmetric encryption schemes is defined via the following game:

(1) $(s, m_0, m_1) \leftarrow A_1(1^t)$.
(2) $b \leftarrow \{0,1\}$.
(3) $k \leftarrow \mathbb{K}_{\mathtt{SK}}(t)$.
(4) $c^* \leftarrow \mathbb{E}_{\mathtt{SK}}(k, m_b)$.
(5) $b' \leftarrow A_2^{\mathcal{O}}(c^*, s)$.

In the above s is state information.

We let $\mathcal{O}$ denote an oracle to which the adversary has access. There are two main possibilities for this oracle depending on the attack model for our game:

- PA Model: In this passive attack model the adversary does not have access to any oracles.
- CCA Model: In this model the oracle $\mathcal{O}_k$ is a decryption oracle for the key k chosen by the challenger in the third step of the above game. This oracle is only available in the second stage of A's game and it is not allowed to be called on the challenge ciphertext c^*.

If we let MOD denote either PA or CCA, the adversary's advantage in the game, called Find-Guess or FG, is defined to be

$$\mathrm{Adv}^{\mathtt{FG-MOD}}_{\mathtt{SK}}(A) = |2\Pr[b' = b] - 1|.$$

A one-time symmetric encryption scheme is considered to be secure, in the sense of a given attack model if, for all PPT adversaries, the advantage in the above game is a negligible function of the security parameter t.

For our purposes we will require a one-time symmetric encryption scheme that is secure in the sense of FG-CCA. We call such a *data encapsulation mechanism* (DEM). The construction of DEMs from PA secure symmetric encryption schemes and MACs is discussed in [8].

3.1.4. *Hybrid Constructions.* We state the theorem for an identity based KEM-DEM construction from [2]. We assume that the key space output by the KEMs corresponds to the key space required by the DEM. Our construction follows that in [8, Section 7.3] and consists of the natural concatenation of the key encapsulation

followed by the data encapsulation of the message under the key encapsulated by the first component. We denote such a ciphertext $C = (c_1, c_2)$ henceforth, where c_1 encapsulates the key and c_2 encapsulates the data, and we refer to such a construction as *hybrid*.

The following theorem is a natural generalisation of Theorem 5 of [8] and is taken from [2]. Note that, unlike the equivalent result in [8], we are implicitly assuming that for all keys, all encapsulations decapsulate properly. It would be straightforward to generalise the result; however, the soundness condition that we are assuming applies to all the primitives that we consider here.

Theorem 1. *Let A be a PPT adversary against the hybrid ID-based encryption scheme in the sense of ID-IND-CCA2 adversaries, then there exists PPT adversaries B_1 and B_2, whose running time is essentially that of A, such that*

$$\mathrm{Adv}_{\mathrm{ID}}^{\mathtt{ID-IND-CCA}}(A) \leq 2\mathrm{Adv}_{\mathrm{ID-KEM}}^{\mathtt{ID-IND-CCA}}(B_1) + \mathrm{Adv}_{\mathrm{DEM}}^{\mathtt{FG-CCA}}(B_2).$$

3.2. **Security Results.** In [6] the security of our scheme is proved in two steps. In the first step a simpler ID-OW-CPA IBE scheme is proved secure; one should think of this as an analogue of the BasicIdent in [4]. The second step is to apply the generic construction of Bentahar *et al.* from [2] of an IND-ID-CCA2 ID-KEM from an ID-OW-CPA IBE scheme.

The final security result is relative to the following problem.

Definition 1 (q-Bilinear Diffie-Hellman Inverse (q-BDHI) [3])**.** *Given group elements $(g_1, g_2, g_2^x, g_2^{x2}, \ldots, g_2^{x^q})$ with $x \in_R \mathbb{Z}_p$, compute $\hat{e}(g_1, g_2)^{1/x}$.*

The proof requires that there is an isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ with $\psi(g_2) = g_1$ which can be computed by the simulator, who is using the adversary to solve the q-BDHI problem. The details of the proof may be found in [6]; here we just state the result:

Theorem 2. *If A is a PPT algorithm that breaks the SK-KEM construction above using a chosen ciphertext attack, then there exists a PPT algorithm B, with*

$$\begin{aligned}\mathrm{Adv}_{\mathrm{ID-KEM}}^{\mathtt{ID-IND-CCA2}}(A) \quad \leq \quad & 2 \cdot q' \cdot \left((q \cdot q_2 + 1) \cdot \mathrm{Adv}^{q-\mathtt{BDHI}}(B) + \frac{1}{2^n} + \frac{q+1}{p} \right) \\ & + \frac{2q_D}{p},\end{aligned}$$

where q_1, q_2, q_3, q_4, q_X and q_D are the number of queries made by A to H_1, H_2, H_3, H_4, the key extraction oracle and the decryption oracle respectively, $q = q_1 + q_X + 1$ and $q' = q_3 + q_4 + q_D$.

We only use Theorem 2 to validate the design of the SK-KEM; we do not use the specifics of the reduction to inform the choice of key size.

3.3. **Recommended Key Sizes.** In Table 6 we recommend parameter sizes for SK-KEM. We include three options that we believe correspond to the security offered by DEMs based on AES-128, AES-192 and AES-256, respectively. The size of the groups used is denoted by p and k is used for the *embedding degree* (the the group $\mathbb{G}_T$ is *embedded* in th multiplicative group of the finite field $\mathbb{F}_{p^k}$). These recommendations follow the recommendation from NIST for comparing ECC and RSA key sizes [18].

TABLE 6. Recommended parameter sizes associated with given AES key sizes

	n	$\log_2 p \geq$	$k \log_2 p \geq$	Hash Function
AES-128	128	256	3072	SHA-256
AES-192	192	384	8192	SHA-384
AES-256	256	512	15360	SHA-512

4. Known limitations and disadvantages

The scheme's only limitations are theoretical. The security proof is strictly speaking only valid in the random oracle model, and the security reduction is to a "non-standard" problem, i.e. q-BDHI. However, given the performance advantage we do not see this as a major problem. For example, no one "in the real world" suggests one should use Cramer-Shoup as an encryption scheme as opposed to something like ECIES, simply because the former has a proof in the standard model relative to a standard assumption, whilst the latter has a proof in the random oracle model relative to the "non-standard" Gap-DH problem.

There has been recent concern expressed about the q-strong DH problem [7] and its variants such as q-BDHI. However, one should note that the q-strong DH and the static-DH problem [5] are closely related and hence the new results are not surprising at all.

5. Intellectual property issues

To the best of our knowledge the basic SK-KEM scheme has no patent issues, excluding the usual ones related to possible implementation tricks. As mentioned earlier a specific design goal was to be patent free, hence none of our institutions (Identum, Hewlett-Packard, Katholieke Universiteit Leuven, Royal Holloway, Universidade do Minho, University of Bristol, and University of Middlesex) have filed any patent applications on the basic SK-KEM scheme.

References

[1] M. Barbosa and P. Farshim. Efficient identity-based key encapsulation to multiple parties. In *Cryptography and Coding, 2005*, Springer-Verlag LNCS 3796, 428–441, 2005.

[2] K. Bentahar, P. Farshim, J. Malone-Lee and N.P. Smart. Generic constructions of identity-based and certificateless KEMs. Cryptology ePrint Archive, Report 2005/058. 2005.

[3] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *Advances in Cryptology - Eurocrypt 2004*, Springer-Verlag LNCS 3027, 223–238, 2004.

[4] D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In *Advances in Cryptology - Crypto 2001*, volume 2139 of LNCS, pages 213-229. Springer-Verlag, 2001.

[5] D.R.L. Brown and R.P. Gallant. The static Diffie–Hellman problem. Cryptology ePrint Archive, Report 2004/306.

[6] L. Chen, Z. Cheng, J. Malone-Lee and N.P. Smart. An efficient ID-KEM based on the Sakai-Kasahara key construction. *IEE Proceedings, Information Security*, **153**, 19–26, 2006.

[7] J.H. Cheon. Security analysis of the strong Diffie–Hellman problem. In *Advances in Cryptology – Eurocrypt 2006*, Springer-Verlag LNCS 4004, 1–11, 2006.

[8] 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**, 167-226, 2004.

[9] A. Dent. A designer's guide to KEMs. In *Cryptography and Coding, 2003*, Springer-Verlag LNCS 2898, 133–151, 2003.
[10] C. Gentry. Practical identity-based encryption without random oracles. In *Advances in Cryptology – Eurocrypt 2006*, Springer-Verlag LNCS 4004, 445–464, 2006.
[11] F. Hess, N.P. Smart and F. Vercauteren. The Eta pairing revisited. Cryptology ePrint Archive, Report 2006/110.
[12] S. Galbraith, K. Paterson and N.P. Smart. Pairings for cryptographers. In preparation.
[13] R. Granger, D. Page and N.P. Smart. High security pairing-based cryptography revisited. To appear *Algorithmic Number Theory Symposium – ANTS VII*, 2006.
[14] Institute of Electrical and Electronics Engineers Standards Association. *IEEE 1363–2000: Standard Specifications for Public Key Cryptography*, 2000.
[15] International Organization for Standardization. *ISO/IEC 18033–4, Information technology - Security techniques - Encryption algorithms - Part 2: Asymmetric Ciphers*, 2005.
[16] International Organization for Standardization. *ISO/IEC 18033–4, Information technology — Security techniques — Encryption Algorithms — Part 4: Stream Ciphers*, 2005.
[17] E. Kiltz. Chosen-ciphertext secure identity-based encryption in the standard model with short ciphertexts. Preprint, 2006.
[18] National Institute of Standards and Technology. NIST Recommendation for Key Management Part 1: General, NIST Special Publication 800-57. August, 2005. Available from `http://csrc.nist.gov/publications/nistpubs/800-57/SP800-57-Part1.pdf`.
[19] R. Sakai and M. Kasahara. ID based cryptosystems with pairing on elliptic curve. Cryptology ePrint Archive, Report 2003/054. 2003.
[20] N.P. Smart and F. Vercauteren. On computable isomorphisms in efficient pairing based systems. Cryptology ePrint Archive, Report 2005/116. 2005.
[21] B. R. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology – EUROCRYPT – 2005*, Springer-Verlag LNCS 3494, 114–127, 2005.

Departmento de Informática, Universidade do Minho, Campus de Gaultar, 4710-057 Braga, Portugal.
E-mail address: mbb@di.uminho.pt

Hewlett-Packard Laboratories, Filton Road, Stoke Gifford, Bristol, BS34 8QZ, United Kingdom.
E-mail address: liqun.chen@hp.com

School of Computing Science, Middlesex University, White Hart Lane, London, N17 8HR, United Kingdom.
E-mail address: m.z.cheng@mdx.ac.uk

Identum Ltd, The Quad, Stubbings Estate, Henley Road, Maidenhead, Berkshire, SL6 6QL, United Kingdom.
E-mail address: mark.chimley@identum.com

Information Security Group, Royal Holloway, University of London, Egham Hill, Egham, Surrey, TW20 0EX United Kingdom.
E-mail address: a.dent@rhul.ac.uk

Department of Computer Science, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, United Kingdom.
E-mail address: farshim@cs.bris.ac.uk

Hewlett-Packard Laboratories, Filton Road, Stoke Gifford, Bristol, BS34 8QZ, United Kingdom.
E-mail address: keith.harrison@hp.com

Department of Computer Science, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, United Kingdom.
E-mail address: malone@cs.bris.ac.uk

Department of Computer Science, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, United Kingdom.
E-mail address: nigel@cs.bris.ac.uk

COSIC - Electrical Engineering, Katholieke Universiteit Leuven, Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium.
E-mail address: fvercaut@esat.kuleuven.ac.be