Multi-receiver Identity-Based Key Encapsulation with Shortened Ciphertext

Sanjit Chatterjee[1] and Palash Sarkar[2]

[1] Indian Institute of Science Education and Research
HC VII, Sector III, Salt Lake City
(IIT Kharagpur Kolkata Campus)
India 700106
[2] Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108
palash@isical.ac.in

Abstract. This paper describes two identity based encryption (IBE) protocols in the multi-receiver setting. The first protocol is secure in the selective-ID model while the second protocol is secure in the full model. The proofs do not depend on the random oracle heuristic. The main interesting feature of both protocols is that the ciphertext size is $|S|/N$, where S is the intended set of receivers and N is a parameter of the protocol. To the best of our knowledge, in the multi-receiver IBE setting, these are the first protocols to achieve sub-linear ciphertext sizes. There are three previous protocols for this problem – two using the random oracle heuristic and one without. We make a detailed comparison to these protocols and highlight the advantages of the new constructions.

Keywords: Multi-receiver encryption, identity based encryption, bilinear pairing.

1 Introduction

In a multi-recipient public key encryption scheme [4,21,5] all users use a common public key encryption system. Suppose there are n users indexed by $1, \ldots, n$; user i having public and private key pair (pk_i, sk_i). A sender who wants to send messages $M_1, \ldots, M_n$ to users $1, \ldots, n$, where M_i is intended for the user i, encrypts M_i using pk_i and sends the resulting ciphertexts $C_1, \ldots, C_n$. This general setting is referred to as multi-plaintext, multi-recipient public key encryption scheme in the literature [21]. If a single message is encrypted, i.e., $M_1 = \cdots = M_n = M$, then we get a single-plaintext, multi-recipient public key encryption scheme. In terms of functionality the later is same as a public key broadcast encryption [17,18].

Alternatively, one can send an encapsulated session key K to multiple parties, whereas the original message M is encrypted through a symmetric encryption scheme using K. In this case, the ciphertext consists of the encapsulation of K,

R. Barua and T. Lange (Eds.): INDOCRYPT 2006, LNCS 4329, pp. 394–408, 2006.

together with an encryption of M using K. Smart [24] considered this notion of mKEM, i.e., an efficient key encapsulation mechanism for multiple parties in the KEM-DEM philosophy.

In the identity-based setting [22,9], the public key corresponding to each user is her/his identity. Given an identity v, a trusted private key generator (PKG) creates the secret key corresponding to v using its own master secret. Now consider the problem of encrypting the same message M for a large set of identities, for example in a group mail. One can either directly encrypt M or use a key-encapsulation mechanism. A trivial solution would be to encrypt (resp. encapsulate) M (resp. K) separately for each individual identities and then transmit them separately. Let the set of identities be S. Then one has to perform $|S|$ many independent encryptions/encapsulation, where $|S|$ denotes the cardinality of S. This solution is clearly too expensive in terms of bandwidth requirement as well as pairing computation.

Baek, Safavi-Naini and Susilo considered this problem in [1]. Along with a formal definition and security model for MR-IBE, they proposed a construction based on the Boneh-Franklin IBE using bilinear pairing. This protocol was proved secure in the selective-ID model *using* the random oracle heuristic. Independent of this work, Barbosa and Farshim [2] proposed an identity-based key encapsulation scheme for multiple parties. This is an extension of the concept of mKEM of Smart [24] to the identity-based setting. Their construction was inspired by the "OR" construction of Smart for access control [23] using bilinear pairing. Security of this scheme also uses the random oracle heuristic, though in the full model. A construction without using the random oracle heuristic has been described in [14]. The construction is based on the Boneh-Boyen (H)IBE [6].

Our contribution: One common limitation of all the above protocols – be it encryption or key encapsulation and whether they use the random oracle heuristic or not – is that the ciphertext size becomes large as the set S of intended recipients increase. For all three protocols, the ciphertext consists of approximately $|S|$ many elements of an elliptic curve group of suitable order.

The context of the current work is based on the following scenario.

- The sender uses a broadcast channel for transmission. Each receiver picks out the part relevant to him/her from the entire broadcast.
- Each recipient gets to know the entire set of receivers. In other words, each receiver knows who are the other persons receiving the same message.

In a broadcast transmission, it is of interest to lower the amount of data to be transmitted. Secondly, since each receiver gets to know the entire set of receivers, this set has to be broadcast along with the message. Thus, the only way of reducing the amount of transmission is by reducing the size of the ciphertext (or the encapsulation of the secret key).

In this work, we concentrate on the problem of reducing the ciphertext size in multi-receiver identity based key encapsulation (mID-KEM). We give constructions where the expected ciphertext size is a fraction of $|S|$. This, comes at

a cost of increasing the private key size. In other words, what we achieve is a controllable trade-off between the ciphertext size and the private key size.

Our first construction is proved secure in the selective-ID model while the second construction is secure in the full model. Both the protocols are proved to be secure *without* using the random oracle heuristic. Also, for both the protocols, we first prove security against chosen plaintext attacks and then adapt the techniques of Boyen-Mei-Waters [11] to attain CCA-security.

Our technique for constructing the mID-KEM is based on the constant size ciphertext hierarchical identity based encryption (HIBE) protocol (BBG-HIBE) in [8]. The algebraic ideas behind the construction are drawn from this work though there are a few differences to be taken care of. Some of these are discussed below. First, an mID-KEM does not require the key delegation property of a HIBE. Though this is not directly required, the simulation of key-extraction queries in the security proof has to use the techniques from [8]. Second, in a HIBE an encryption is to an identity whose maximum length is equal to the depth of the HIBE. In an mID-KEM, the set of users to which a key has to be encapsulated can be large and has no relation to the public parameters of the system. The third difference is that while the security of our protocol depends on the hardness of the DBDHE problem introduced in [8], the security of the BBG-HIBE itself can be based on the weaker problem wDBDHI* (see the full version of [8] at the eprint server).

Due to lack of space, the proofs are omitted. These are available in the full version of the paper at the eprint sever maintained by IACR.

2 Definitions

2.1 Cryptographic Bilinear Map

Let G_1 and G_2 be cyclic groups of same prime order p and $G_1 = \langle P \rangle$, where we write G_1 additively and G_2 multiplicatively. A mapping $e : G_1 \times G_1 \rightarrow G_2$ is called a cryptographic bilinear map if it satisfies the following properties:

- Bilinearity: $e(aP, bQ) = e(P, Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in \mathbb{Z}_p$.
- Non-degeneracy: If $G_1 = \langle P \rangle$, then $G_2 = \langle e(P, P) \rangle$.
- Computability: There exists an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

Since $e(aP, bP) = e(P, P)^{ab} = e(bP, aP)$, $e()$ also satisfies the symmetry property. Modified Weil pairing [9] and Tate pairing [3,19] are examples of cryptographic bilinear maps where G_1 is an elliptic curve group and G_2 is a subgroup of a finite field. This motivates our choice of the additive notation for G_1 and the multiplicative notation for G_2. In papers on pairing implementation [3,19], it is customary to write G_1 additively and G_2 multiplicatively. Initial "pure" protocol papers such as [9,20] followed this convention. Later works such as [6,7,25] write both G_1 and G_2 multiplicatively. Here we follow the first convention as it is closer to the known examples of bilinear maps.

2.2 mID-KEM Protocol

Following [1,2], we define a multi-receiver Identity-Based Key Encapsulation (mID-KEM) scheme as a set of four algorithms: Setup, Key Generation, Encapsulation and Decapsulation.

Setup: It takes input 1^κ, where κ is a security parameter and returns the system public parameters together with the master key. The system parameters include a description of the groups G_1, G_2 and $e()$. These are publicly known while the master key is known only to the private key generator (PKG).

Key Generation: It takes as input an identity v, the system public parameters and returns a private key d_{v}, using the master key. The identity v is used as a public key while d_{v} is the corresponding private key.

Encapsulation: It takes as input the public parameters and a set of identities S and produces a pair (K, Hdr), where K is a key for a symmetric encryption algorithm and Hdr is a header which encapsulates K. An actual message M is encrypted by a symmetric encryption algorithm under K to obtain C_M. The actual broadcast consists of (S, Hdr, C_M), where (S, Hdr) is called the full header and C_M is called the broadcast body.

Decapsulation: It takes as input a pair (S, Hdr); an identity v and a private key d_{v} of v. If $\mathsf{v} \in S$, then it returns the symmetric key K which was used to encrypt the message. This K can be used to decrypt the broadcast body C_M to obtain the actual message M.

2.3 Hardness Assumption

Security of our mID-KEM scheme is based on the *decisional bilinear Diffie-Hellman exponent* (DBDHE) problem introduced by Boneh-Boyen-Goh in [8]. The l-DBDHE problem is stated as follows.

> Given $P, Q, aP, \ldots, a^{l-1}P, a^{l+1}P, \ldots, a^{2l}P$ for random $a \in \mathbb{Z}_p$ and $Z \in G_2$, decide whether $Z = e(P,Q)^{a^l}$ or whether Z is random.

Let $\mathcal{B}$ be a probabilistic algorithm which takes this instance as input and produces a bit as output. The advantage of $\mathcal{B}$ in solving this decision problem is defined to be

$$\mathsf{Adv}_{\mathcal{B}}^{\mathsf{DBDHE}} = |\mathsf{Pr}[\mathcal{B}(P, Q, \overrightarrow{R}, e(P,Q)^{a^l}) = 1] - \mathsf{Pr}[\mathcal{B}(P, Q, \overrightarrow{R}, Z) = 1]|$$

where $\overrightarrow{R} = (aP, a^2P, \ldots a^{l-1}P, a^{l+1}P, \ldots, a^{2l}P)$ and Z is a random element of G_2. The probability is calculated over the random choices of $a \in \mathbb{Z}_p$, $Z \in G_2$ and also the random bits used by $\mathcal{B}$.

The (t, ϵ, l)-DBDHE assumption holds if $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{DBDHE}} \leq \epsilon$ for any algorithm $\mathcal{B}$ for the l-DBDHE problem, where the runtime of $\mathcal{B}$ is at most t.

2.4 Security Model of mID-KEM

We define indistinguishability under chosen ciphertext attack for multi-receiver identity-based key encapsulation scheme. The adversarial behaviour is defined by the following game between an adversary $\mathcal{A}$ and a simulator $\mathcal{B}$.

$\mathcal{A}$ is allowed to query two oracles – a decryption oracle and a key-extraction oracle. At the initiation, it is provided with the system public parameters.

Phase 1: $\mathcal{A}$ makes a finite number of queries where each query is addressed either to the decryption oracle or to the key-extraction oracle. In a query to the decryption oracle, it provides the full broadcast header as well as the identity under which it wants the decryption. In return, the simulator $\mathcal{B}$ provides $\mathcal{A}$ with either the corresponding symmetric key or bad. In a query to the key-extraction oracle, it provides an identity and the corresponding private key is given to it by the simulator $\mathcal{B}$. $\mathcal{A}$ is allowed to make these queries adaptively, i.e., any query may depend on the previous queries as well as their answers.

Challenge: At this stage, $\mathcal{A}$ fixes a set of identities S^*, under the (obvious) constraint that it has not asked for the private key of any identity in S^*. The simulator $\mathcal{B}$ generates a proper pair (K^*, Hdr^*) corresponding to the set of identities S^* as defined by the encryption algorithm. It then chooses a random bit γ and sets $K_0 = K^*$ and sets K_1 to be a random symmetric key of equal length. $\mathcal{B}$ returns $(K_\gamma, \mathsf{Hdr}^*)$ to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ now issues additional queries just like Phase 1, with the (obvious) restriction that it cannot ask the decryption oracle for the decryption of Hdr^* under any identity in S^* nor the key-extraction oracle for the private key of any identity in S^*.

Guess: $\mathcal{A}$ outputs a guess γ' of γ.

Adversarial Advantage: The advantage of the adversary $\mathcal{A}$ in attacking the mID-KEM scheme is defined as:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{mID\text{-}KEM}} = |\mathsf{Pr}[(\gamma = \gamma')] - 1/2|.$$

The mID-KEM protocol is said to be $(\epsilon, t, q_{\mathsf{V}}, q_C, \sigma)$-secure against chosen ciphertext attacks (IND-mID-CCA secure), if for any adversary running in time t; making q_{V} key-extraction queries; making q_C decryption queries; and with $|S^*| \leq \sigma$, we have $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{mID\text{-}KEM}} \leq \epsilon$.

We include the upper bound on the size of set of target identities S^* as part of the adversary's resources. The set S^* is the set of identities under which the adversary wants the encryption in the challenge stage. Intuitively, increasing the size of this set allows the possibility of the adversary receiving more information.

2.5 Selective-ID Model

We can have a weaker version of the above model by restricting the adversary. In this model, the adversary has to commit to the set of target identities S^* even before the protocol is set-up. During the actual game, it cannot ask the key-extraction oracle for the private key of any identity in S^* and in the challenge stage the set S^* is used to generate K^*.

Henceforth, we will call this restricted model the selective-ID (sID) model and the unrestricted model to be the full model.

2.6 CPA Security

We may impose another restriction on the adversary, namely, we do not allow the adversary access to the decryption oracle. This restriction can be made on both the full and the sID models. One can define the advantage of such an adversary in a manner similar to above.

As above, a mID-KEM protocol is said to be (ϵ, t, q, σ) IND-mID-CPA secure if for any adversary running in time t; making q queries to the key-extraction oracle; and with $|S^*| \leq \sigma$, we have $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{mID\text{-}KEM}} \leq \epsilon$.

In the identity based setting, there are generic methods [13,10] for transforming a CPA-secure protocol into a CCA-secure protocol. Thus, one can simply prove the CPA-security of a protocol and then apply known transformations to obtain CCA-security. Recently [11], a non-generic method has been obtained for transforming the CPA-secure protocols in [25,6] into CCA-secure protocols.

3 Construction of CPA-Secure mID-KEM in the sID Model

Let, $e(), G_1, G_2$ be as defined in Section 2.1 and identities are assumed to be elements of $\mathbb{Z}_p^*$.

Setup: Let P be a generator of G_1. Choose a random secret $x \in \mathbb{Z}_p$ and set $P_1 = xP$. Also choose random elements P_2, P_3 in G_1 and a random vector $\overrightarrow{U} = (U_1, \ldots, U_N)$ with entries in G_1. The significance of the parameter N is discussed later. The public parameters are

$$\langle P, P_1, P_2, P_3, \overrightarrow{U}, H() \rangle$$

where $H()$ is a publicly computable surjective function $H : \mathbb{Z}_p^* \to \{1, \ldots, N\}$.

The master secret key is xP_2.

Key Generation: Given any identity $\mathsf{v} \in \mathbb{Z}_p^*$, this algorithm generates the private key d_{v} of v as follows.

Compute $k = H(\mathsf{v}) \in \{1, \ldots, N\}$. Choose a random element $r \in \mathbb{Z}_p$ and output

$$d_{\mathsf{v}} = (d_0, d_1, b_1, \ldots, b_{k-1}, b_{k+1}, \ldots, b_N)$$

where $d_0 = (xP_2 + r(P_3 + \mathsf{v}U_k))$; $d_1 = rP$; and for $1 \leq i \leq N$, $b_i = rU_i$. Note that b_k is not part of the private key for any identity v, such that $k = H(\mathsf{v})$. The private key for any identity consists of $(N + 1)$ elements of G_1.

A Notation: For a set S of identities, we introduce the notation

$$V(S) = P_3 + \sum_{\mathsf{v} \in S} \mathsf{v}U_{H(\mathsf{v})}. \tag{1}$$

The expression $\mathsf{v}U_{H(\mathsf{v})}$ denotes the scalar multiplication of U_k by v, where $k = H(\mathsf{v})$. The use of this notation will simplify the description of the protocol.

Encapsulation: Let S be a set of identities for which we want to encapsulate a session key K. We partition S into several subsets in the following manner.

Let $H(S) = \{j_1, \ldots, j_k\}$ be the set of distinct indices obtained by applying $H()$ to the elements of the set S. For $1 \leq i \leq k$, let $\{s_{i,1}, \ldots, s_{i,\tau_i}\}$ be the subset of all elements in S which map to j_i. Let $\tau = \max_{1 \leq i \leq k}(\tau_i)$. We view S as a (possibly incomplete) $k \times \tau$ matrix having entries $s_{i,j}$ where $1 \leq i \leq k$ and $1 \leq j \leq \tau_i$. For $1 \leq j \leq \tau$, define the set S_j to be the jth column of this matrix. Then S is a disjoint union of $S_1, \ldots, S_\tau$ and for all j, $|S_j| = |H(S_j)|$, i.e., H is injective on S_j.

Choose a random $s \in \mathbb{Z}_p$, compute $K = e(P_1, P_2)^s$ and then set the header as

$$\mathsf{Hdr} = (sP, sV(S_1), \ldots, sV(S_\tau)).$$

The header consists of $\tau + 1$ elements of G_1. The full header is the tuple $(S_1, \ldots, S_\tau, \mathsf{Hdr})$, where K is used to obtain the broadcast body C_M by encrypting the message M using symmetric encryption. The entire broadcast consists of $(S_1, \ldots, S_\tau, \mathsf{Hdr}, C_M)$.

Discussion

1. There is no security assumption on $H()$. We need $H()$ to be surjective to ensure that the entire set of public parameters are used. While there is no security requirement on $H()$, we still expect the output of $H()$ to be uniformly distributed. The expected size of the header is equal to $|S|/N$ under this assumption. We would like to emphasize that $H()$ is *not* assumed to be a random oracle. In particular, this assumption is *not* used in the security proof. Rather, this should be seen as the usual assumption on a hash function used in data/file structure.
2. The parameter N in the protocol controls the trade-off between header size and the size of the public parameters as well as the size of the private key. For each i in the range $\{1, \ldots, N\}$, there is a component of the public key corresponding to this index i. Any identity is mapped to an index in the range $\{1, \ldots, N\}$ using $H()$ and the corresponding component of the public key is used to obtain the component d_0 of the private key of the identity. If S is a random set of identities, then under the assumption that the output of $H()$ is uniformly distributed, the expected size of the header is $1 + \lceil |S|/N \rceil$. N still has another role – in the security reduction this is the maximum number of identities that the adversary is allowed to target.

Decapsulation: An individual user does not need the full header

$$(S_1, \ldots, S_\tau, \mathsf{Hdr} = (C_0, C_1, \ldots, C_\tau))$$

for decapsulation. For a user with identity v, it is sufficient for him to obtain (S_j, C_0, C_j), such that $\mathsf{v} \in S_j$. This can be easily picked out by the user from the general broadcast of the full header. Note that by construction, for $\mathsf{v}, \widehat{\mathsf{v}} \in S_j$ with $\mathsf{v} \neq \widehat{\mathsf{v}}$, we have $H(\mathsf{v}) \neq H(\widehat{\mathsf{v}})$, i.e., in other words $H()$ is injective on S_j.

Thus, the input to the decapsulation algorithm is a tuple $(\widehat{\mathsf{v}}, S, C, D)$, where $\widehat{\mathsf{v}} \in S$; $H()$ is injective on S; $C = sP$ and $D = sV(S)$. The private key $d_{\widehat{\mathsf{v}}}$ for $\widehat{\mathsf{v}}$ is

$$d_{\widehat{\mathsf{v}}} = (d_0, d_1, b_1, \ldots, b_{k-1}, b_{k+1}, \ldots, b_N)$$

where $k = H(\widehat{\mathsf{v}})$. Suppose that r was used during the generation of the private key $d_{\widehat{\mathsf{v}}}$ for $\widehat{\mathsf{v}}$. Then $d_0 = xP_2 + r(P_3 + \widehat{\mathsf{v}}U_{H(\widehat{\mathsf{v}})}$, $d_1 = rP$ and $b_i = rU_i$. Compute

$$\begin{aligned}
\mathsf{key}_{\widehat{\mathsf{v}}} &= d_0 + \sum_{\mathsf{v} \in S \wedge \mathsf{v} \neq \widehat{\mathsf{v}}} \left(\mathsf{v} b_{H(\mathsf{v})}\right) \\
&= xP_2 + r(P_3 + \widehat{\mathsf{v}}U_{H(\widehat{\mathsf{v}})}) + r \sum_{\mathsf{v} \in S \wedge \mathsf{v} \neq \widehat{\mathsf{v}}} \left(\mathsf{v} U_{H(\mathsf{v})}\right) \\
&= xP_2 + r\left(P_3 + \sum_{\mathsf{v} \in S} \mathsf{v} U_{H(\mathsf{v})}\right).
\end{aligned}$$

It then obtains the session key K from the components C_0, C_j of Hdr, $\mathsf{key}_{\widehat{\mathsf{v}}}$ and d_1 as

$$\frac{e(\mathsf{key}_{\widehat{\mathsf{v}}}, C_0)}{e(C_j, d_1)} = \frac{e\left(xP_2 + r(P_3 + \sum_{\mathsf{v} \in S} \mathsf{v} U_{H(\mathsf{v})}), sP\right)}{e\left(s(P_3 + \sum_{\mathsf{v} \in S} \mathsf{v} U_{H(\mathsf{v})}), rP\right)} = e(P_1, P_2)^s.$$

In the above protocol, each identity is mapped to an index in the range $\{1, \ldots, N\}$ using the function $H()$. Each index has an associated public parameter. While generating the private key for an identity, the identity itself as well as the public parameter part corresponding to the index of this identity is "mixed" to the master secret xP_2, while the parts corresponding to all other public parameters are provided individually to the users. In the case, where two identities have the same index, the corresponding private keys will be different. One reason for this is that the randomizers r will be different for the two identities. More importantly, the first component of the private key depends on the actual value of the identity and hence will be different for the two distinct identities.

Now suppose that during encryption, an identity v in the broadcast set S has an index i. Then as shown above, the entity possessing a private key corresponding to v can decrypt the message. On the other hand, suppose there is an identity $\mathsf{v}' \notin S$ such that $H(\mathsf{v}) = i = H(\mathsf{v}')$. Then we must be assured that possessing the private key of v' does not allow decryption. Suppose that during encryption v is assigned to set S_j. Then the $(j+1)$th entry of the header is

of the form $s\sum_{\mathsf{v}\in S_j} V_{\mathsf{v}}$. In particular, this sum includes the value V_{v}. During decryption, v constructs $\mathsf{key}_{\mathsf{v}} = xP_2 + r\sum_{\mathsf{v}\in S_j} V_{\mathsf{v}}$, where r was used to generate the private key for v. Let $S'_j = (S_j \setminus \{\mathsf{v}\}) \cup \{\mathsf{v}'\}$. Then v' can compute the value $\mathsf{key}_{\mathsf{v}'} = xP_2 + r'\sum_{\mathsf{v}\in S'_j} V_{\mathsf{v}}$, where r' was used to generate the private key for v'. In general, there is no way to compute $\mathsf{key}_{\mathsf{v}}$ from $\mathsf{key}_{\mathsf{v}'}$.

Theorem 1. *The mID-KEM protocol is (t, q, ϵ, N)-secure against chosen plaintext attacks in the sID model under the assumption that $(t+t', \epsilon', N+1)$-DBDHE assumption holds for $\langle G_1, G_2, e()\rangle$, where $\epsilon \leq \epsilon'$ and t' is the time required for $O(q)$ scalar multiplications in G_1 and $O(N)$ multiplications in $\mathbb{Z}_p$.*

4 Construction of CPA-Secure mID-KEM in the Full Model

We first discuss the modifications required in the construction of Section 3 to make it secure against adaptive adversary. Here, identities are assumed to be n-bit strings. Let ℓ be a size parameter, $1 < \ell \leq n$, chosen a-priori.

Set-up: The public parameter consists of N vectors, each of length ℓ. Let, $\overrightarrow{P_3} = (P_1, \ldots, P_N)$ and $\mathcal{U} = (\overrightarrow{U_1}, \ldots, \overrightarrow{U_N})$, where $\overrightarrow{U_i} = (U_{i,1}, \ldots, U_{i,\ell})$ with each P_i and each $U_{i,j}$ in G_1. So the public parameters are

$$\langle P, P_1, P_2, \overrightarrow{P_3}, \mathcal{U}, H()\rangle.$$

The function $H : \{0,1\}^n \to \{1, \ldots, N\}$ is a surjective map whose role is as in the protocol of Section 3.

A Notation: For any identity $\mathsf{v} = (\mathsf{v}_1, \ldots, \mathsf{v}_\ell)$, where each v_i is a bit string of length n/ℓ, we define

$$\left.\begin{array}{ll} V(\mathsf{v}) = P_{3,k} + \sum_{j=1}^{\ell} \mathsf{v}_j U_{k,j} & \text{where } k = H(\mathsf{v}); \\ V(S) = \sum_{\mathsf{v}\in S} V(\mathsf{v}) & \text{for any set } S \text{ of identities.} \end{array}\right\} \tag{2}$$

When the context is clear we use V_{v} in place of $V(\mathsf{v})$.

The parameters (N, n, ℓ) control the configuration of the mID-KEM. Hence, we will refer to the construction as (N, n, ℓ) mID-KEM construction.

Key Generation: Given any identity v this algorithm generates the private key d_{v} of v as follows. Compute $k = H(\mathsf{v}) \in \{1, \ldots, N\}$. Choose a random element $r \in \mathbb{Z}_p$ and output

$$d_{\mathsf{v}} = (d_0, d_1, a_1, \ldots, a_{k-1}, a_{k+1}, \ldots, a_N, \overrightarrow{b_1}, \ldots, \overrightarrow{b_{k-1}}, \overrightarrow{b_{k+1}}, \ldots, \overrightarrow{b_N})$$

where $d_0 = (xP_2 + rV_{\mathsf{v}})$; $d_1 = rP$; and for $1 \leq i \leq N$, $a_i = rP_{3,i}$; $\overrightarrow{b_i} = r\overrightarrow{U_i}$. Note that a_k and $\overrightarrow{b_k}$ are not part of the private key for any identity v, such that $k = H(\mathsf{v})$. The private key for any identity consists of $N(\ell+1) - (\ell-1)$ elements of G_1.

Encapsulation: Let S be the set of identities under which we want to perform the encryption. Form the sets $S_1, \ldots, S_\tau$ based on the function $H()$ as in the encapsulation algorithm of the protocol in Section 3. Choose a random $s \in \mathbb{Z}_p$. The header consists of

$$\mathsf{Hdr} = (sP, sV(S_1), \ldots, sV(S_\tau)).$$

The full header is formed from Hdr as in Section 3.

Decapsulation: As in Section 3, a user with identity $\widehat{\mathsf{v}} = (\widehat{\mathsf{v}_1}, \ldots, \widehat{\mathsf{v}_\ell})$ does not require the full header for decapsulation. From the full header, the user forms the tuple $(\widehat{\mathsf{v}}, S, sP, sV(S))$. The user also has the private key $d_{\widehat{\mathsf{v}}}$ corresponding to $\widehat{\mathsf{v}}$. Note that $\widehat{\mathsf{v}} = (\widehat{\mathsf{v}}_1, \ldots, \widehat{\mathsf{v}}_\ell)$ and suppose r was used to generate $d_{\widehat{\mathsf{v}}}$. Recall that $d_0 = xP_2 + r\left(P_{3,H(\widehat{\mathsf{v}})} + \sum_{j=1}^{\ell} \widehat{\mathsf{v}_j} U_{H(\widehat{\mathsf{v}}),j}\right)$. Compute

$$\mathsf{key}_{\widehat{\mathsf{v}}} = d_0 + \sum_{\mathsf{v} \in S, \mathsf{v} \neq \widehat{\mathsf{v}}} \left(a_{H(\mathsf{v})} + \sum_{j=1}^{\ell} \mathsf{v}_j U_{H(\mathsf{v}),j} \right).$$

It can be shown that $\mathsf{key}_{\widehat{\mathsf{v}}} = xP_2 + rV(S)$. Using $\mathsf{key}_{\widehat{\mathsf{v}}}$, it is possible to obtain $e(P_1, P_2)^t$ as in Section 3.

The following theorem shows that the above scheme is secure against an adaptive adversary.

Theorem 2. *The (N, n, ℓ) mID-KEM protocol is (ϵ, t, q, h)-secure against chosen plaintext attacks in the full model under the assumption that $(t', \epsilon', (N+1))$-DBDHE assumption holds, where*

$$\epsilon \leq 2(2\sigma(\mu_l + 1))^h \epsilon'; \text{ and } t' = t + \chi + O(\epsilon^{-2} \ln(\epsilon^{-1}) \lambda^{-1} \ln(\lambda^{-1}))$$

with $\mu_l = l((2^n)^{1/l} - 1)$, $\lambda = 1/(2(2\sigma(\mu_l + 1))^h)$, $\sigma = max(2q, 2^{n/\ell})$ and χ is the time required for $O(q)$ scalar multiplications in G_1 and $O(N)$ multiplications in $\mathbb{Z}_p$.

5 Security Against Chosen Ciphertext Attack

We adapt the technique of Boyen-Mei-Waters [11] for attaining CCA-security. This technique applies to both the sID-secure and the full model secure protocols giving rise to the following two protocols.

1. $\mathbf{M}_1$: A CCA-secure mID-KEM in the sID model.
2. $\mathbf{M}_2$: A CCA-secure mID-KEM in the full model.

Below we describe how to obtain $\mathbf{M}_1$ by modifying the construction in Section 3. Essentially the same thing also holds for the full model secure protocol.

Set-Up: In addition to the set-up for the CPA-secure scheme, we choose a random element W from G_1. This W is part of the public parameters. There is also an injective encoding $H_1 : G_1 \to \mathbb{Z}_p$.

Key Generation: Remains same as before.

Encapsulation: In addition to what was provided earlier, one more element B is provided as part of the header. This element is computed in the following manner. Let the full header for the CPA-secure protocol be $(S_1, \ldots, S_\tau, sP, sA_1, \ldots, sA_\tau)$. Let $\nu = H_1(sP)$ and set $B = s(\nu P_2 + W)$. The full header for the new protocol is

$$(S_1, \ldots, S_\tau, sP, sA_1, \ldots, sA_\tau, B).$$

Note that one B is used for all the users, i.e., B is independent of the identity.

Decapsulation: The input to the decapsulation algorithm is a tuple $(\mathsf{v}, S, C = sP, D = sA, B)$. The portion (v, S, sP, sA) was used as input to the decapsulation algorithm for the CPA-secure protocol. The new entry is B. Compute $A = V(S) = P_3 + \sum_{\mathsf{v} \in S} \mathsf{v} U_{H(\mathsf{v})}$, $\nu = H_1(C)$ and $E = \nu P_2 + W$. The following public verification checks are performed:

1. Does $\mathsf{v} \in S$?
2. Is H injective on S?
3. Is $e(P, D) = e(C, A)$?
4. Is $e(P, B) = e(C, E)$?

If the answer to any of the above questions is no, then return bad. Otherwise, obtain the encapsulated key as in the case of the CPA-secure algorithm.

6 Comparison

Here we make a comparison of $\mathbf{M}_1$ and $\mathbf{M}_2$ with the previous three protocols proposed so far, i,e., BSS [1], BaFa [2] and CS [14] protocols.

We would like to note that the various protocols are based on different hardness assumptions. Strictly speaking, a comparison between them is not possible. The comparison we make assumes that all the hard problems are "equally hard". Under this assumption, one can perhaps make a meaningful comparison.

For the sake of uniformity we assume that the BSS protocol is used for key encapsulation and the header (Hdr) consists of only elements of G_1. During decryption the symmetric key (K) is decapsulated from this header as in our construction of Section 3. The BSS protocol was proved secure in the sID model using the random oracle assumption. This protocol is in a sense an extension of the Boneh-Franklin IBE [9] to the multi-receiver setting. Here the public parameter consists of three elements of G_1, including the generator of the group. The private key corresponding to a given identity is generated using a map-to-point function modeled as a random oracle and consists of a single element of G_1.

The protocol of [14] is based on the Boneh-Boyen HIBE [6]. The protocol is proved to be CPA-secure but can be made CCA-secure using a technique similar to the one described in this paper (based on the Boyen-Mei-Waters [11] construction). The protocol does not use the random oracle heuristic. Here the public parameter consist of $4 + N$ elements of G_1, where N is the number of

target identities committed by the adversary [14]. The private key is similar to that in the Boneh-Boyen scheme [6] and consists of two elements of G_1.

In Table 1, we make a comparison between the public parameter size, private key size and the header size for the three protocols. The public parameter size and the private key size are smaller for the BSS protocol – that is because they *use* a random oracle to generate the private key. For both CS as well as $\mathbf{M}_1$ the public parameter is linear in N, where N is a parameter of the model and once chosen is constant. Also, the private key size for the new construction is N, while that for CS is only 2. The real advantage of the new protocol is in shortening the header size. The header size for both BSS and CS protocols are $|S| + 1$. In contrast, the expected header size for the new protocol is $\lceil |S|/N \rceil + 1$.

A typical value of N would be 16. For this value of N, the public parameter of $\mathbf{M}_1$ consists of 19 elements (elliptic curve points) of G_1 and the private key consists of 16 elements of G_1. Both these sizes are within acceptable limits. On the other hand, a broadcast to a group of around 1000 users will consist of around 60 elements of G_1, which is also within reasonable limits. On the other hand, for both the BSS and the CS protocols, the broadcast will consist of 1000 elements of G_1, which can be prohitively costly. Thus, in certain situations, the reduction in the size of the broadcast can be more significant than the increases in the sizes of the public parameters and that of the private key.

Efficiency: Let us consider the efficiency of different operations (key generation, encapsulation and decapsulation) for the various protocols. The efficiency of key generation is proportional to the size of the private key for all the protocols.

Similarly, the decapsulation efficiency is proportional to the part of the full header required by an individual user for decapsulation. For M_1, this is N scalar multiplications plus two pairing computations, while for the other protocols this is only two pairing computations. The reason is that in attempting to reduce the header size below $|S|$, we are grouping users during encapsulations. Thus, during decapsulation, each user has to use the information about which of the other groups have been grouped with it. Since each group has at most N users, decapsulation requires at most N scalar multiplications.

The encapsulation efficiency for the BSS protocol is $1 + |S|$ scalar multiplications. On the other hand, the CS protocol requires $N \times |S|$ scalar multiplications. Thus, the cost of removing the random oracle heuristic is a decrease in the efficiency of encapsulation. This cost is even more than the trivial protocol of encrypting separately to all the identities in S using (say) the BB-IBE. The cost in this case will be $2|S|$ scalar multiplications. However, the header size in this trivial protocol is going to be $2|S|$. If we want to reduce the header size to $|S|$ (and avoid the random oracle heuristic), then the CS protocol increases the cost of encapsulation.

The encapsulation efficiency for $\mathbf{M}_1$ is approximately $|S|(N + 1)/N$. For $N > 1$, this value is less than $2|S|$ and hence our protocol is more efficient than the trivial protocol. On the other hand, it is less efficient than the BSS protocol. Thus, compared to the BSS protocol, our contribution is to decrease the header size and avoid the random oracle heuristic.

Table 1. Comparison of mID-KEM protocols secure in the selective-ID model. Here S is the set of identities to which the encapsulation is formed.

Protocol	Hardness Assumption	Random Oracle	Pub. Para. size (elts. of G_1)	Pvt. Key size (elts. of G_1)	Header size (elts. of G_1)
BSS [1]	DBDH	Yes	3	1	$\|S\| + 1$
CS [14]	DBDH	No	$4 + N$	2	$\|S\| + 1$
$\mathbf{M}_1$	DBDHE	No	$3 + N$	N	$\lceil \|S\|/N \rceil + 1$

Table 2. Comparison of mID-KEM protocols secure in the full model. The security degradation of both protocols is exponential in N, which is the number of attacked users. Here S is the set of identities to which the encapsulation is formed.

Protocol	Hardness Assumption	Random Oracle	Pub. Para. size (elts. of G_1)	Pvt. Key size (elts. of G_1)	Header size (elts. of G_1)
BaFa [2]	GBDH	Yes	2	1	$\|S\| + 2$
$\mathbf{M}_2$	DBDHE	No	$3 + N + N\ell$	$N(\ell + 1) - (\ell - 1)$	$\lceil \|S\|/N \rceil + 1$

Full Model Secure Protocols: The comparison of the full model secure protocols is given in Table 2. The security of the BaFa protocol was proved in the full model using the random oracle heuristic assuming the hardness of gap bilinear Diffie-Hellman problem. The system suffers from an exponential security degradation which is around q^N, where q is the number of random oracle queries and N is the number of target identities. The construction of Section 4 is also secure in the full model but *without* using the random oracle heuristic. The security degradation is again q^N. The large security degradation of both these protocols imply that these protocols are not really useful when N is around 10 or so. Thus, our contribution in the full model is really to show that it is possible to obtain security without using random oracle and simultaneously obtain sublinear header size.

7 Conclusion

We present two protocols for multi-receiver identity-based key encapsulation system. The first protocol is secure in the selective-ID model while the second protocol is secure in the full model. The security proofs do not use the random oracle heuristic and are based on the hardness of decisional bilinear Diffie-Hellman exponentiation problem. The main advantage of the new protocols over the previous ones is that for encryption to a set S of users, the header size in the new protocols is sub-linear in $|S|$, whereas for the previous protocols this is approximately $|S|$.

Acknowledgement. We would like to thank the reviewers for their comments and for pointing out several typos.

References

1. Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Efficient Multi-receiver Identity-Based Encryption and Its Application to Broadcast Encryption. In Serge Vaudenay, editor, *Public Key Cryptography*, volume 3386 of *Lecture Notes in Computer Science*, pages 380–397. Springer, 2005.
2. M. Barbosa and P. Farshim. Efficient identity-based key encapsulation to multiple parties. In Nigel P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *Lecture Notes in Computer Science*, pages 428–441. Springer, 2005.
3. Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 354–368. Springer, 2002.
4. Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *EUROCRYPT*, pages 259–274, 2000.
5. Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemeas. In Desmedt [16], pages 85–99.
6. Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In Cachin and Camenisch [12], pages 223–238.
7. Dan Boneh and Xavier Boyen. Secure Identity Based Encryption Without Random Oracles. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer, 2004.
8. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. In Cramer [15], pages 440–456. Full version available at Cryptology ePrint Archive; Report 2005/015.
9. Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. Earlier version appeared in the proceedings of CRYPTO 2001.
10. Dan Boneh and Jonathan Katz. Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption. In Alfred Menezes, editor, *CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2005.
11. Xavier Boyen, Qixiang Mei, and Brent Waters. Direct Chosen Ciphertext Security from Identity-Based Techniques. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 320–329. ACM, 2005.
12. Christian Cachin and Jan Camenisch, editors. *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*. Springer, 2004.
13. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In Cachin and Camenisch [12], pages 207–222.
14. Sanjit Chatterjee and Palash Sarkar. Generalization of the Selective-ID Security Model for HIBE Protocols. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 241–256. Springer, 2006. Revised version available at Cryptology ePrint Archive, Report 2006/203.
15. Ronald Cramer, editor. *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*. Springer, 2005.

16. Yvo Desmedt, editor. *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, volume 2567 of *Lecture Notes in Computer Science*. Springer, 2002.
17. Yevgeniy Dodis and Nelly Fazio. Public Key Broadcast Encryption for Stateless Receivers. In Joan Feigenbaum, editor, *Digital Rights Management Workshop*, volume 2696 of *Lecture Notes in Computer Science*, pages 61–80. Springer, 2002.
18. Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In Desmedt [16], pages 100–115.
19. Steven D. Galbraith, Keith Harrison, and David Soldera. Implementing the Tate Pairing. In Claus Fieker and David R. Kohel, editors, *ANTS*, volume 2369 of *Lecture Notes in Computer Science*, pages 324–337. Springer, 2002.
20. Craig Gentry and Alice Silverberg. Hierarchical ID-Based Cryptography. In Yuliang Zheng, editor, *ASIACRYPT*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer, 2002.
21. Kaoru Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography*, volume 2274 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 2002.
22. Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.
23. Nigel P. Smart. Access control using pairing based cryptography. In Marc Joye, editor, *CT-RSA*, volume 2612 of *Lecture Notes in Computer Science*, pages 111–121. Springer, 2003.
24. Nigel P. Smart. Efficient Key Encapsulation to Multiple Parties. In Carlo Blundo and Stelvio Cimato, editors, *SCN*, volume 3352 of *Lecture Notes in Computer Science*, pages 208–219. Springer, 2004.
25. Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In Cramer [15], pages 114–127.