Variants of Waters' Dual-System Primitives Using Asymmetric Pairings

Somindu C. Ramanna[1], Sanjit Chatterjee[2], and Palash Sarkar[1]

[1] Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108.
e-mail: {somindu_r,palash}@isical.ac.in

[2] Department of Computer Science and Automation
Indian Institute of Science
Bangalore, India 560012.
e-mail: sanjit@csa.iisc.ernet.in

Abstract. Waters, in 2009, introduced an important technique, called dual-system encryption, to construct identity-based encryption (IBE) and related schemes. The resulting IBE scheme was described in the setting of symmetric pairing. A key feature of the construction is the presence of random tags in the ciphertext and decryption key. Later work by Lewko and Waters has removed the tags and proceeding through composite-order pairings has led to a more efficient dual-system IBE scheme using asymmetric pairings whose security is based on non-standard but static assumptions. In this work, we have systematically simplified Waters 2009 IBE scheme in the setting of asymmetric pairing. The simplifications retain tags used in the original description. This leads to several variants, the first one of which is based on standard assumptions and in comparison to Waters original scheme reduces ciphertexts and keys by two elements each. Going through several stages of simplifications, we finally obtain a simple scheme whose security can be based on two standard assumptions and a natural and minimal extension of the decision Diffie-Hellman problem for asymmetric pairing groups. The scheme itself is also minimal in the sense that apart from the tags, both encryption and key generation use exactly one randomiser each. This final scheme is more efficient than both the previous dual-system IBE scheme in the asymmetric setting due to Lewko and Waters and the more recent dual-system IBE scheme due to Lewko. We extend the IBE scheme to hierarchical IBE (HIBE) and broadcast encryption (BE) schemes. Both primitives are secure in their respective full models and have better efficiencies compared to previously known schemes offering the same level and type of security.

Keywords: identity-based encryption, dual-system encryption, asymmetric pairing.

1 Introduction

Constructions of identity based encryption schemes constitute one of the most challenging problems of public-key cryptography. The notion of IBE was proposed in [16] and solved in [3, 7]. This lead to a great deal of research on the topic. The solution in [3], though simple and elegant, had several features which were not satisfactory from a theoretical point of view.

In this work, we will be interested in IBE schemes built from bilinear pairings. Till date, most pairing based cryptographic schemes have been based on a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}$ is a prime-order group of elliptic curve points over a finite field and $\mathbb{G}_T$ is a subgroup of a finite field. Such maps arise from Weil and Tate pairings and there is an extensive literature on efficient implementation of such maps. Since the two components of the domain of e are same, such an e is called a symmetric pairing. Another kind of pairings, where the order of $\mathbb{G}$ is composite has been proposed [4]. Such pairings are called composite-order pairings and provide additional flexibility in designing schemes. The trade-off, however, is that computing the pairing itself becomes significantly slower and also the representation of the group elements becomes substantially longer.

Symmetric pairings (over prime order groups), are neither the most general nor the most efficient of possible pairings over elliptic curves. A general bilinear map is of the form $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1$ is a prime-order group of points of an elliptic curve over a finite field $\mathbb{F}$ and $\mathbb{G}_2$ is a group (of the same prime-order) of points of the same curve over an extension of $\mathbb{F}$. Such maps are called asymmetric pairings. Studies [19, 8, 5] have indicated that compared to symmetric pairings, asymmetric pairings are much faster and more compact to implement.

An important work on pairing based IBE is [20] which builds upon earlier work in [1, 2] to provide an efficient IBE scheme with several nice theoretical properties. Variants have been reported [6, 14] which result in IBE schemes which are efficient and have practical sized parameters. Though important, a drawback of the scheme in [20] is that the size of the public parameters grows linearly with the security parameter.

In a major innovation, Waters [21] introduced a new technique – called dual-system encryption – for construction of IBE schemes and related primitives. The scheme presented in [21] has the feature that the size of the public parameters is constant while retaining the other nice theoretical properties of the previous scheme also due to Waters [20]. Dual-system encryption is by itself an interesting notion and worthy of further investigation. The goal of a better understanding of dual-system encryption would be to obtain IBE schemes with improved efficiency compared to the one proposed in [21].

An immediate follow-up work [13] took the route of composite-order pairings. Such pairing groups have 'more structure' which can possibly help in getting a clearer understanding of the technique. (Waters remarks that his scheme [21] was first obtained for composite order groups.) The approach taken by [13] is to look at a realization of the IBE scheme of [1] in the setting of composite order groups so as to obtain adaptive-id security. They also gave a conversion of their composite-order IBE scheme to an IBE scheme using prime-order asymmetric pairing. In a very recent work [12], the framework of dual-system encryption has been thoroughly investigated and an IBE scheme using prime-order pairing has been presented. We note that the conversion from composite-order to prime-order pairings in [13] and considering prime order groups in [12] are motivated by efficiency considerations.

Waters IBE scheme in [21] is based on symmetric pairings. The security of the scheme is based on the hardness of the decision linear (DLin) and the decision bilinear Diffie-Hellman (DBDH) assumptions. It is of interest to convert this to asymmetric pairings. For one thing, this will enable faster and smaller implementations which will arise from the advantages of asymmetric pairings over their symmetric variants. There is, however, another reason. Use of asymmetric pairings brings forward the possibility of reducing the number of group elements in the ciphertext and keys. In fact, Waters [21] himself mentions: "using the SXDH assumption we might hope to shave off three group elements from both ciphertexts and private keys". The rationale for this comment is that for asymmetric pairings, the decision Diffie-Hellman (DDH) assumption holds for both $\mathbb{G}_1$ and $\mathbb{G}_2$. This is the symmetric external Diffie-Hellman (SXDH) assumption. For symmetric pairings the DDH assumption does not hold in $\mathbb{G}$. Using the SXDH assumption will potentially lead to a simpler scheme requiring a lesser number of group elements.

Following up on the above mentioned remark by Waters, we have systematically investigated the various possibilities for using asymmetric pairings. To start the study, we performed a straightforward conversion to the setting of asymmetric pairings. The scheme in [21] is quite complex. Several scalars are used in the public parameters, encryption and key generation. These have definite and inter-connected roles in the security proof. Our first task was to pin down the relationships between these scalars and separate them out. This enabled us to work with one group of scalars with minimal changes to other groups.

With a good understanding of the roles of the scalars, we are able to apply simplifications in a stage-wise manner. The first simplification gives an IBE scheme (Scheme 1) which reduces ciphertexts and keys by two elements each and whose security can be based on DDH1 (DDH assumption in $\mathbb{G}_1$), DLin and the DBDH assumptions. We argue that the DDH2 assumption cannot be directly used. So, the afore-mentioned suggestion by Waters cannot be fulfilled. On the other hand, we show that using a natural and minimal extension of the DDH2 assumption, a significantly more efficient scheme (Scheme 6) can be obtained.

Waters original scheme [21] used random tags in the ciphertext and the decryption key. Simplification of this scheme by both Lewko-Waters [13] and Lewko [12] yielded IBE schemes which did not use such tags. In contrast, all our simplifications retain the tags used in the original description [21]. Even then, we are able to obtain significant simplifications and efficiency improvements. This suggests that for the purpose of simplification as an IBE it is not important to do away with the tags. Removing them has other positive consequences such as obtaining a constant size ciphertext hierarchical IBE [13].

An interesting feature about Scheme 6 is the minimal use of randomisers. Apart from the tags, exactly one randomiser each is used for encryption and key generation. Since at least one randomiser each for encryption and key generation will be required for any secure IBE scheme, this feature leads us to believe that there cannot be any further simplification of Waters scheme in [21] while retaining the tags. To show that our simplification retains the flexibility of the original technique by Waters, we present an analogue of the HIBE scheme along with a security proof in Section G. This HIBE scheme inherits all the nice theoretical properties from [21], but, provides improved efficiency. From this HIBE scheme we construct an adaptively secure BE scheme which is more efficient than all the previously known BE schemes with adaptive security. The construction is given in Section 4.2 and the security proof in Section H.

A comparison of the features of various IBE schemes based on the dual-system technique is shown in Tables 1 and 2. The columns $\#\mathcal{PP}$, $\#\mathcal{MSK}$, #cpr, #key provide the number of group elements in the public parameters, the master secret key, ciphertexts and decryption keys. The public parameter and ciphertexts consist of elements of $\mathbb{G}_1$ while the master secret key and decryption keys consist of elements of $\mathbb{G}_2$. Encryption efficiency counts the number of scalar multiplications in $\mathbb{G}_1$ while decryption efficiency counts the number of pairings that are required. Key generation (a less frequent activity) efficiency is given by the number of scalar multiplications in $\mathbb{G}_2$. Currently, Scheme 6 is the most efficient among all the known dual-system IBE schemes.

scheme	$\#\mathcal{PP}$	$\#\mathcal{MSK}$	#cpr	#key	enc eff	dec eff	key gen	assump
Waters-09 [21]	13	5	9	8	14	9	12	DLin, DBDH
Lewko-11 [12]	24	30	6	6	24	6	6	DLin
Scheme 1	9	8	7	6	10	7	9	DDH1, DLin, DBDH

Table 1. Comparison of dual-system IBE schemes secure under standard assumptions. Waters-09 and Lewko-11 are originally described using symmetric pairings; the figures are for direct conversion to asymmetric pairings. .

scheme	$\#\mathcal{PP}$	$\#\mathcal{MSK}$	#cpr	#key	enc eff	dec eff	key gen	assump
LW [13]	9	6	6	6	9	6	10	LW1, LW2, DBDH
Scheme 6	6	7	4	4	7	5	6	DDH1, DDH2v, DBDH

Table 2. Comparison of dual-system IBE schemes secure under non-standard but static assumptions. Note that DDH1 is a weaker assumption than LW1 and DDH2v is a weaker assumption than LW2. .

2 Prerequisites

Definitions of IBE, HIBE and the corresponding security models is given in Section A. Here we briefly describe asymmetric pairings and related assumptions. For more details on these the reader is referred to [19, 8, 5].

2.1 Bilinear maps

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order p. $\mathbb{G}_1$ and $\mathbb{G}_2$ are written additively while $\mathbb{G}_T$ is written multiplicatively. A cryptographic bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ has the following properties.

1. **Bilinearity:** For $A_1, B_1 \in \mathbb{G}_1$ and $A_2, B_2 \in \mathbb{G}_2$, $e(A_1 + B_1, A_2) = e(A_1, A_2)e(B_1, A_2)$ and $e(A_1, A_2 + B_2) = e(A_1, A_2)e(A_1, B_2)$.
2. **Non-degeneracy:** For generators P_1 of $\mathbb{G}_1$ and P_2 of $\mathbb{G}_2$, $e(P_1, P_2) \neq 1_T$, where 1_T is the identity element of $\mathbb{G}_T$.
3. **Efficiency:** The map e is efficiently computable.

A bilinear map is called symmetric or a Type-1 bilinear map if $\mathbb{G}_1 = \mathbb{G}_2$; otherwise it is asymmetric. Asymmetric bilinear maps are further classified into Type-2 and Type-3 bilinear maps. In the Type-2 setting, there is an efficiently computable isomorphism either from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$ whereas in the Type-3 setting no such isomorphisms are known. Previous works [19, 8, 5] have established that the Type-3 setting is the most efficient from an implementation point of view.

We introduce some notation: Given generators P_1 of $\mathbb{G}_1$ and P_2 of $\mathbb{G}_2$ and elements $R_1 \in \mathbb{G}_1$ and $R_2 \in \mathbb{G}_2$, the notation $R_1 \sim R_2$ indicates that R_1 has the same discrete logarithm to base P_1 as that of R_2 to base P_2. For a set $\mathbb{X}$, let $x \in_{\mathrm{R}} \mathbb{X}$ denote that x is a uniform random element of $\mathbb{X}$.

In the following, we will assume the availability of an asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ where $\mathbb{G}_1 = \langle P_1 \rangle$, $\mathbb{G}_2 = \langle P_2 \rangle$ and both $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of the same prime order p. Being of prime order, any non-identity element of $\mathbb{G}_1$ is a generator of the group and the same holds for $\mathbb{G}_2$.

2.2 Hardness Assumption

We introduce a new hardness assumption for Type-3 pairings. Here we provide a discussion of this. Section B describes the other standard hardness assumptions required in this work.

Let P_1 and P_2 be random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The DDH problem in $\mathbb{G}_1$ is to decide, given $(P_1, x_1P_1, x_2P_1, P_2, Z_1)$, whether $Z_1 = x_1x_2P_1$ or Z_1 is a random element of $\mathbb{G}_1$. Here $x_1, x_2 \in_R \mathbb{Z}_p$. Similarly one can define the DDH assumption in $\mathbb{G}_2$. In this case, an instance has the form $(P_1, P_2, x_1P_2, x_2P_2, Z_2)$ and the task is to determine whether $Z_2 = x_1x_2P_2$ or whether Z_2 is a random element of $\mathbb{G}_2$. For convenience we will denote the DDH problem in $\mathbb{G}_1$ as DDH1 and that in $\mathbb{G}_2$ as DDH2. The symmetric external Diffie-Hellman (SXDH) assumption is that both DDH1 and DDH2 problems are hard. Note that for symmetric pairing (i.e., for $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G} = \langle P \rangle$), DDH is easy to solve by comparing $e(P, Z_2)$ with $e(x_1P, x_2P)$.

We will use DDH1 in our proofs. But DDH2 is not directly applicable to our proofs. An instance of DDH2 has a single element P_1 of $\mathbb{G}_1$. For our proofs, we will require some information about x_1P_1 to be carried as part of the instance. If the instance is directly augmented by x_1P_1, then the problem becomes easy, since one can compute the two pairings $e(x_1P_1, x_2P_2)$ and compare to $e(P_1, Z_2)$. Suppose that instead of x_1P_1 we include the elements zP_1 and zx_1P_1 where z is chosen randomly from $\mathbb{Z}_p$. This pair of elements carries some information about x_1P_1, but, not the element itself. An instance will now be $(P_1, zP_1, zx_1P_1, P_2, x_1P_2, x_2P_2, Z_2)$. It, however, is easy to check whether Z_2 equals $x_1x_2P_2$ by checking whether $e(zx_1P_1, x_2P_2)$ equals $e(zP_1, Z_2)$. This suggests that the information about zP_1 itself needs to be blinded by another randomiser. So, instead of having zP_1 directly, the elements dP_1, dzP_1 and dP_2 are included where d is a random element of $\mathbb{Z}_p$. The information about x_1P_1 is carried by the elements dP_1, dzP_1, zx_1P_1 and dP_2. Augmenting an instance of DDH2 with these elements embeds information about x_1P_1 but, does not seem to provide any way to use this information to determine whether Z_2 is real or random. The entire thing can now be formulated as an assumption in the following manner.

Assumption DDH2v. Let P_1,P_2 be random generators of $\mathbb{G}_1$,$\mathbb{G}_2$ respectively and let $x_1, x_2, d, z \in_R \mathbb{Z}_p$. The problem is to decide, given $(P_1, dP_1, dzP_1, zx_1P_1, P_2, dP_2, x_1P_2, x_2P_2, Z_2)$, whether $Z_2 = x_1x_2P_2$ or Z_2 is a random element of $\mathbb{G}_2$.

This corresponds to a two-level blinding of x_1P_1. We have seen that providing x_1P_1 directly or using a single-level blinding makes the problem easy. So, a two-level blinding is the minimum that one has to use to get to an assumption about hardness.

The assumption DDH2v (the "v" stands for variant) is no harder than DDH2. This is because an instance of DDH2v contains an embedded instance of DDH2 and an algorithm to solve DDH2 can be invoked on this embedded instance to solve the instance of DDH2v. On the other hand, there is no clear way of using an algorithm to solve DDH2v to solve DDH2. Intuitively, this is due to the fact that an instance of DDH2v contains some information about x_1P_1 whereas an instance of DDH2 does not contain any such information.

In our reduction, we will use the assumption DDH2v. Since assumption DDH2v does not appear earlier in the literature, it is a non-standard assumption. Having said this, we would also like to remark that DDH2v arises naturally as a minimal assumption when one tries to augment an instance of DDH2 with some information about x_1P_1 while maintaining the hardness of the problem. A proof of security of this assumption in the generic group model is provided in Section B.1. We feel that assumption DDH2v will have applications elsewhere for schemes based on asymmetric pairings.

3 Framework for Conversion

Our goal is to transform Waters-2009 IBE scheme to the asymmetric setting so that we can reduce the number of components both in the ciphertext and the key. To that end, we first perform a straightforward conversion of Waters IBE from the setting of symmetric pairing to the setting of asymmetric pairing. (See [21] for the original description of Waters 2009 scheme.)

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a Type 3 bilinear map and let P_1 and P_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. After the conversion, either the ciphertext or the key will consist of elements of $\mathbb{G}_1$; the other will consist of elements from $\mathbb{G}_2$. Elements of $\mathbb{G}_1$ will have shorter representation compared to those of $\mathbb{G}_2$. For encryption, we want the ciphertext to be short and hence we choose its elements to be from $\mathbb{G}_1$. The public parameters will consist of elements of $\mathbb{G}_1$ whereas the master secret key will consist of elements of $\mathbb{G}_2$. We note that if the final goal were to construct a signature scheme, then one would perform a conversion where the master secret key consists of elements of $\mathbb{G}_1$.

A straightforward conversion will have the same structure as the one described in [21]. We use the convention in this and later schemes that the subscript 1 will denote elements of $\mathbb{G}_1$ while the subscript 2 will denote elements of $\mathbb{G}_2$. Further, messages are elements of $\mathbb{G}_T$ and identities are elements of $\mathbb{Z}_p$.

To generate $\mathcal{PP}$, first choose α, b, a_1, a_2 randomly from $\mathbb{Z}_p$ and consider the following. Let v, v' and v'' be random elements of $\mathbb{Z}_p$ and define $V_2 = vP_2$, $V_2' = v'P_2$ and $V_2'' = v''P_2$. Let $\tau = v + a_1 v'$ and $\tau' = v + a_2 v''$. Set $T_1 = \tau P_1$ and $T_1' = \tau' P_1$. The $\mathcal{PP}$ will have elements $Q_1, U_1, W_1 \in \mathbb{G}_1$ and correspondingly the master secret key will have elements $Q_2, U_2, W_2 \in \mathbb{G}_2$ with $Q_2 \sim Q_1$, $U_2 \sim U_1$ and $W_2 \sim W_1$. The structure of the $\mathcal{PP}$ and the $\mathcal{MSK}$ are as follows.

$\mathcal{PP}$: $(P_1, bP_1, a_1P_1, a_2P_1, ba_1P_1, ba_2P_1, T_1, T_1', bT_1, bT_1', Q_1, W_1, U_1)$.
$\mathcal{MSK}$: $(P_2, \alpha P_2, \alpha a_1 P_2, V_2, V_2', V_2'', Q_2, W_2, U_2)$.

Encrypt$(M, \mathsf{id}, \mathcal{PP})$: Randomisers $s_1, s_2, t, \mathsf{ctag}$ are chosen from $\mathbb{Z}_p$ and define $s = s_1 + s_2$. The ciphertext is $(C_0, C_1, \ldots, C_7, E_1, E_2, \mathsf{ctag})$ where the various elements are defined as follows.

$$C_0 = M \cdot e(P_1, P_2)^{ba_1\alpha s_2}$$
$$C_1 = bsP_1,\ C_2 = ba_1s_1P_1,\ C_3 = a_1s_1P_1,\ C_4 = ba_2s_2P_1,$$
$$C_5 = a_2s_2P_1, C_6 = s_1T_1 + s_2T_1',\ C_7 = s_1bT_1 + s_2bT_1' - tW_1$$
$$E_1 = t(\mathsf{id}Q_1 + \mathsf{ctag}W_1 + U_1),\ E_2 = tP_1$$

KeyGen$(\mathsf{id}, \mathcal{MSK}, \mathcal{PP})$: Randomisers $r_1, r_2, z_1, z_2, \mathsf{ktag}$ are chosen from $\mathbb{Z}_p$ and define $r = r_1 + r_2$. The key $\mathcal{SK}_{\mathsf{id}}$ is $(K_1, \ldots, K_7, \mathsf{ktag})$ where the various elements are defined as follows.

$$K_1 = \alpha a_1 P_2 + rV_2,\ K_2 = -\alpha P_2 + rV_2' + z_1P_2,\ K_3 = -bz_1P_2,\ K_4 = rV_2'' + z_2P_2,$$
$$K_5 = -bz_2P_2,\ K_6 = r_2bP_2,\ K_7 = r_1P_2$$
$$D = r_1(\mathsf{id}Q_2 + \mathsf{ktag}W_2 + U_2).$$

The decryption algorithm requires 9 pairings and succeeds only if ctag in the ciphertext is not equal to ktag of the decryption key, an event which occurs with overwhelming probability. See [21] for the details.

Waters defines algorithms to generate semi-functional ciphertexts and keys. These cannot be computed without knowledge of the secret components and are only used in the security reduction. They are defined such that one should be able to decrypt a semi-functional ciphertext with a normal key and a normal ciphertext with a semi-functional key; but decryption of a semi-functional ciphertext with a semi-functional key should fail.

Semi-functional Ciphertext: Let $C'_0, \ldots, C'_7, E'_1, E'_2$, ctag be ciphertext elements normally generated by the **Encrypt** algorithm for message M and identity id. Choose $\mu \in \mathbb{Z}_p$ at random. Let $V'_1 = v'P_1$ and $V''_1 = v''P_1$ so that $V'_1 \sim V'_2$ and $V''_1 \sim V''_2$. The semi-functional ciphertext generation algorithm will modify the normal ciphertext as: $C_0 = C'_0$, $C_1 = C'_1$, $C_2 = C'_2$, $C_3 = C'_3$, $E_1 = E'_1$, $E_2 = E'_2$ and

$$C_4 = C'_4 + ba_2\mu P_1,\ C_5 = C'_5 + a_2\mu P_1,\ C_6 = C'_6 - a_2\mu V''_1,\ C_7 = C'_7 - ba_2\mu V''_1.$$

Semi-functional Key: Let $K'_1, \ldots, K'_7, D'$, ktag be secret key components normally generated by the **KeyGen** algorithm for identity id. Choose at random $\gamma \in \mathbb{Z}_p$. The semi-functional key generation algorithm will modify the normal key as: $K_3 = K'_3$, $K_5 = K'_5$, $K_6 = K'_6$, $K_7 = K'_7$, $D = D'$ and

$$K_1 = K'_1 - a_1a_2\gamma P_2,\ K_2 = K'_2 + a_2\gamma P_2,\ K_4 = K'_4 + a_1\gamma P_2.$$

It is easy to see that one can decrypt a semi-functional ciphertext with a normal key and a normal ciphertext with a semi-functional key. However, decryption of a semi-functional ciphertext with a semi-functional key will fail because the masking factor $e(P_1, P_2)^{ba_1\alpha s_2}$ will be blinded by the factor $e(P_1, P_2)^{ba_1a_2\mu\gamma}$.

Security proof. The security argument for the scheme proceeds through $q + 3$ games where q is the number of key extraction queries made by the adversary. These games are

$$\mathrm{Game}_{real}, \mathrm{Game}_0, \ldots, \mathrm{Game}_q, \mathrm{Game}_{final}.$$

The transition between these games can be seen as three different reductions.

First reduction: The transition from Game_{real} to Game_0 is made by replacing the challenge ciphertext by a semi-functional ciphertext. It is argued that detecting this change should be hard.

Second reduction: There is a sequence of q changes from Game_{k-1} to Game_k (for $k = 1, \ldots, q$). The k-th change is as follows. For the queries numbered 1 to $k - 1$, the adversary is given a semi-functional key; for queries numbered $k + 1$ to q, the adversary is given a normal key. For the k-th query, the adversary is given a response such that deciding whether the response is normal or semi-functional is hard. The challenge ciphertext is semi-functional as in the first reduction.

Third reduction: This tackles the transition from Game_q to Game_{final}. At this point, all responses to key queries are semi-functional and so is the challenge ciphertext. In the last transition, the challenge ciphertext is changed such that deciding whether it is the encryption of a message or whether it is statistically independent of the $\mathcal{PP}$ and the responses is hard.

The first and second reductions are based on the hardness of the DLin problem whereas the third reduction is based on the hardness of the DBDH problem. In the proof, the second reduction is the most complex step. The subtle point is that the simulator should not be able to generate a semi-functional ciphertext for the k-th identity. This is ensured by using algebraic techniques from [1] to create ktag using a pair-wise independent function so that the simulator is able to create a semi-functional ciphertext for id_k only with $\mathsf{ctag} = \mathsf{ktag}$, in which case decryption fails unconditionally.

3.1 An Analysis

Our conversion to asymmetric pairing and subsequent simplifications are based on an analysis of the various scalars used in the scheme and their respective roles in the proof. Based on the scheme itself and a study of the three reductions used by Waters, we make the following observations.

1. $\mathcal{PP}$ uses the scalars a_1, a_2 and b, while $\mathcal{MSK}$ uses the scalars α and a_1.
2. Key generation uses scalar randomisers r_1, r_2 and z_1, z_2. The scalar r is set to $r_1 + r_2$. We will call this the split of r.
3. Ciphertext generation uses the scalar randomisers s_1, s_2 and t. The scalar s is set to $s_1 + s_2$. We will call this the split of s.
4. The first two reductions in Waters proof are based on the DLin assumption. The first reduction uses the split of s whereas the second reduction uses the split of r.

For conversion to asymmetric pairing, the following points are to be noted. These have been inferred from a study of the security proof in [21].

1. The scalar α needs to be retained.
2. There are three basic possibilities for simplifications: remove the split of s; remove the split of r; remove z_1, z_2.
3. Getting rid of a_1 and a_2 and using a single a will eliminate the requirement of the split of s. This also means that the separate z_1 and z_2 are not required and instead a single z can be used.
4. Removing the split of r does not have a direct influence on the other scalars.
5. Removing the split of r and also z_1, z_2 means that the scalar b is no longer required.
6. In all but one of our schemes, the scalar t is kept either as part of the ciphertext or as part of the key. In the final scheme, we show that the scalar t can also be removed. For this scheme, there is a single randomiser s for the ciphertext and a single randomiser r for the key. Note that further reduction in randomness is not possible. Key generation must have at least one randomiser, as otherwise the algorithm becomes deterministic; similarly, encryption must also have at least one randomiser, as otherwise the ciphertext becomes unique.
7. If the split of s and a_1,a_2 are retained, then the first reduction has to be based on DLin. If it is removed, then we can base the first reduction on DDH1.

8. If the split of r is retained, then the second reduction has to be based on DLin. If it is removed, we can no longer base the second reduction on DLin. However, it can neither be based on DDH2 for the following reason. An instance of DDH2 will provide P_1 and some elements of $\mathbb{G}_2$. Apart from P_1 no other element of $\mathbb{G}_1$ is provided. The $\mathcal{PP}$ consists of elements of $\mathbb{G}_1$ which has to be related to the instance in some way. Just having P_1 does not provide any way to construct the $\mathcal{PP}$ in the second reduction. So, removing the split of r implies that the second reduction can be based on neither DLin nor DDH2. The assumption DDH2v introduced in Section 2 provides the necessary mechanism for carrying the proof through.
9. The tags are chosen randomly and they play a crucial role in the security argument. We do not consider removing tags. If the tags are removed, then it will be necessary to introduce copies of the identity-hash (as done in [13]) to obtain the functionality of tags in the semi-functional components. This leads to an increase in the number of elements in the ciphertext and key.

Based on the above points, we explore the different natural ways in which Waters 2009 IBE scheme can be converted to asymmetric pairing. These are discussed below.

Scheme 1: Remove the split of s. This eliminates the requirement of having separate a_1, a_2 and z_1, z_2. Reductions of ciphertext and key are by two elements each. Removing the split of s allows the first reduction to be based on DDH1. Since the split of r is retained, the second reduction is still based on DLin.

Scheme 2. Retain the split of s; this means that separate a_1 and a_2 are required. Remove the split of r and also remove z_1 and z_2; this means that b can be removed. Leads to reductions of ciphertext and key by 3 elements each. The first reduction of the proof can be based on DLin, but, the second reduction cannot be based on either DLin or DDH2.

Scheme 3: Remove the split of s; retain the split of r but, remove z. Reductions of ciphertext and key are by 3 elements each. In the proof, the first reduction can be based on DLin. The second reduction cannot be based on DDH2. Neither can it be based on DLin. This requires a more involved reasoning which we provide in Section F.

Scheme 4: Remove the splits of both r and s, but, retain z. Ciphertext and key are reduced by 3 elements each. In the proof, the first reduction can be based on DDH1, but, the second reduction cannot be based on either DLin or DDH2.

Scheme 5: Remove the splits of both r and s and also remove z. Ciphertext and keys are reduced by 4 elements each. As in the previous case, the first reduction of the proof can be based on DDH1, but, the second reduction cannot be based on either DLin or DDH2.

Scheme 6: In Schemes 1 to 5, the randomiser t is present in the ciphertext. In Scheme 6, the splits of both r and s are removed; z is removed and the role of t is played by s. This leads to a scheme where there is exactly one randomiser for encryption and exactly one randomiser for key generation. Compared to Waters' IBE [21], reduction of the ciphertext is by 5 elements and the reduction of the key is by 4 elements. The first reduction of the proof can be based on DDH1, while the second reduction is based on assumption DDH2v.

In Table 3, we provide the use of scalars in the various schemes. This illustrates the manner in which the simplification has been obtained.

scheme	$\mathcal{PP}$	$\mathcal{MSK}$	key gen	enc
Waters-09 [21]	α, a_1, a_2, b	α, a_1	$r_1, r_2, (r = r_1 + r_2), z_1, z_2$	$s_1, s_2, (s = s_1 + s_2), t$
Scheme 1	α, a, b	α, b	$r_1, r_2, (r = r_1 + r_2), z$	s, t
Scheme 2	α, a_1, a_2	α	r	$s_1, s_2, (s = s_1 + s_2), t$
Scheme 3	α, a, b	α, b	$r_1, r_2, (r = r_1 + r_2)$	s, t
Scheme 4	α, a, b	α, b	r, z	s, t
Scheme 5	α, a	α	r	s, t
Scheme 6	α, a	α	r	s

Table 3. Usage of scalars in various schemes. Note that all the schemes use ktag for key generation and ctag for encryption.

4 Constructions

In this section, we provide the description of Scheme 6. The security proof for this scheme is provided in Appendix E. In Appendix D, we provide the description and security proof for Scheme 1. For Schemes 2 to 5, only the descriptions are provided in Appendix F. These schemes primarily serve the purpose of showing the stepping stones in moving from Scheme 1 to Scheme 6.

4.1 Scheme 6

Descriptions of $\mathcal{PP}$, $\mathcal{MSK}$, ciphertext generation, key generation and decryption are provided.

Let a, v, v' be random elements of $\mathbb{Z}_p$. Set $V_2 = vP_2$, $V_2' = v'P_2$ and $\tau = v + av'$ so that $\tau P_2 = V_2 + aV_2'$.

$\mathcal{PP}$: $(P_1, aP_1, \tau P_1, Q_1, W_1, U_1, e(P_1, P_2)^{\alpha})$.
$\mathcal{MSK}$: $(P_2, \alpha P_2, V_2, V_2', Q_2, W_2, U_2)$.

Encrypt$(M, \mathsf{id}, \mathcal{PP})$: Choose random s, ctag from $\mathbb{Z}_p$; $\mathcal{C}$ is $(C_0, C_1, C_2, C_3, E, \mathsf{ctag})$ where the elements are defined as follows.

$C_0 = M \cdot e(P_1, P_2)^{\alpha s}$,
$C_1 = sP_1,\ C_2 = asP_1,\ C_3 = -\tau sP_1 + sW_1,\ E = s(\mathsf{id}Q_1 + \mathsf{ctag}W_1 + U_1)$.

KeyGen$(\mathsf{id}, \mathcal{MSK}, \mathcal{PP})$: Choose random r, ktag from $\mathbb{Z}_p$; $\mathcal{SK}_{\mathsf{id}}$ is $(K_1, K_2, K_3, D, \mathsf{ktag})$ where the elements are defined as follows.

$K_1 = \alpha P_2 + rV_2,\ K_2 = rV_2',\ K_3 = rP_2,\ D = r(\mathsf{id}Q_2 + \mathsf{ktag}W_2 + U_2)$.

Decrypt $(\mathcal{C}, \mathsf{id}, \mathcal{SK}_{\mathsf{id}}, \mathcal{PP})$: As before, decryption succeeds only when $\mathsf{ctag} \neq \mathsf{ktag}$. Decryption is done in two steps. First compute

$$A_1 = \left(\frac{e(E, K_3)}{e(C_1, D)}\right)^{1/(\mathsf{ctag}-\mathsf{ktag})} = e(W_1, P_2)^{rs}$$

and then $A_2 = e(C_1, K_1)e(C_2, K_2)e(C_3, K_3) = e(P_1, P_2)^{\alpha s}e(W_1, P_2)^{rs}$. Unmask the message as $M = (C_0 \cdot A_1)/A_2$.

Semi-functional ciphertext: Let $(C_0', C_1', C_2', C_3', E', \mathsf{ctag})$ be a normal ciphertext. Choose a random μ from $\mathbb{Z}_p$. The semi-functional ciphertext is $(C_0, C_1, C_2, C_3, E, \mathsf{ctag})$ where $C_0 = C_0'$, $C_1 = C_1'$, $C_2 = C_2' + \mu P_1$, $C_3 = C_3' - \mu V_1'$ and $E = E'$.

Semi-functional key: Let $(K_1', K_2', K_3', D, \mathsf{ktag})$ be a normal key. Choose a random γ from $\mathbb{Z}_p$. The semi-functional key is $(K_1, K_2, K_3, D, \mathsf{ktag})$ where $K_1 = K_1' - a\gamma P_2$, $K_2 = K_2' + \gamma P_2$, $K_3 = K_3'$ and $D = D'$.

The complete security proof for this scheme is given in Section E.

Extension to HIBE: Waters extends the IBE scheme in [21] in a natural way to a HIBE scheme. We show that our simplification of Waters scheme retains the original flexibility. In Section G, we describe a HIBE which extends Scheme 6. This HIBE scheme is secure under the DDH1, DDH2v and the DBDH assumptions and provides lesser and smaller parameters and better efficiencies of key generation, delegation, encryption and decryption compared to the HIBE in [21]. The full security proof for the HIBE is also given in Section G.2.

Conversion to Signature Scheme: There is a "dual" of Scheme 6 where the ciphertext elements are in $\mathbb{G}_2$ and decryption keys consist of elements of $\mathbb{G}_1$. Using Naor's observation, this dual of Scheme 6 can be converted to a secure signature scheme. The signatures will be composed of elements of $\mathbb{G}_1$ and will be smaller than the signatures obtained by the conversion of Waters' 2009 scheme to a signature scheme. In a similar manner, one can convert the dual of the HIBE in Section G to obtain a HIBS scheme where signatures consist elements of $\mathbb{G}_1$.

4.2 Broadcast Encryption

The full version of Waters paper described a broadcast encryption (BE) scheme based on the dual-system IBE in [21]. In this section, we describe a BE scheme based on Scheme 6. The security proof is given in Section H and is based on the hardness of the DDH1, DDH2v and the DBDH problems. The new BE scheme provides adaptive security and is more efficient than previously known BE schemes providing adaptive security [10, 21].

Setup(κ, n): Based on the security parameter κ a group description along with a Type-3 pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is generated. Let n be the total number of users. Generators $P_1 \in_R \mathbb{G}_1$ and $P_2 \in_R \mathbb{G}_2$ are chosen. Also choose random elements $Q_{1,1}, \ldots Q_{1,n}, W_1 \in \mathbb{G}_1$ and $Q_{2,1}, \ldots, Q_{2,n}, W_2 \in \mathbb{G}_2$ such that $Q_{2,i} \sim Q_{1,i}$ for $1 \le i \le n$, $W_2 \sim W_1$. Let α, a, v, v' be random elements of $\mathbb{Z}_p$. Set $V_2 = vP_2$, $V_2' = v'P_2$ and $\tau = v + av'$ so that $\tau P_2 = V_2 + aV_2'$.

$\mathcal{PK}$: $(P_1, aP_1, \tau P_1, Q_{1,1}, \ldots, Q_{1,n}, W_1, e(P_1, P_2)^{\alpha})$.
$\mathcal{MSK}$: $(P_2, \alpha P_2, V_2, V_2', Q_{2,1}, \ldots, Q_{2,n}, W_2)$.

Encrypt$(\mathcal{PK}, S \subseteq \{1, \ldots, n\}, M)$: Choose random s from $\mathbb{Z}_p$; $\mathcal{C}$ for the set S is (C_0, C_1, C_2, C_3, E) where the elements are defined as follows.

$C_0 = M \cdot e(P_1, P_2)^{\alpha s}$,
$C_1 = sP_1$, $C_2 = asP_1$, $C_3 = -\tau s P_1 + sW_1$, $E = s(\sum_{i \in S} Q_{1,i})$.

KeyGen($\mathcal{SK}, j \in \{1, \ldots, n\}$): Choose random r from $\mathbb{Z}_p$; $\mathcal{SK}_j$ is $(K_1, K_2, K_3, D, \forall_{i \neq j} D_i)$ where the elements are defined as follows.

$$K_1 = \alpha P_2 + rV_2,\ K_2 = rV_2',\ K_3 = rP_2$$
$$D = r(Q_{2,j} + W_2),\ D_i = rQ_{2,i} \text{ for } i \neq j$$

Decrypt $(\mathcal{C}, S, \mathcal{SK}_j)$: Decryption works only if $j \in S$. It is done in two steps. First compute

$$A_1 = \left(\frac{e(C_1, D + \sum_{\substack{i \in S \\ i \neq j}} D_i)}{e(E, K_3)} \right) = e(P_1, W_2)^{rs}$$

and then $A_2 = e(C_1, K_1)e(C_2, K_2)e(C_3, K_3) = e(P_1, P_2)^{\alpha s} e(W_1, P_2)^{rs}$. Unmask the message as $M = (C_0 \cdot A_1)/A_2$.

The semi-functional ciphertexts and keys for the BE are defined as follows.

Semi-functional ciphertext: Let $C_0', C_1', C_2', C_3', E$ be ciphertext elements normally generated by the **Encrypt** algorithm for message M and subset $S \subseteq \{1, \ldots, n\}$ of users. Let V_1' be an element of $\mathbb{G}_1$ such that $V_1' \sim V_2'$. Choose $\mu \in \mathbb{Z}_p$ at random. The semi-functional ciphertext generation algorithm will modify the normal ciphertext as: $C_0 = C_0'$, $C_1 = C_1'$, $E = E'$ and

$$C_2 = C_2' + \mu P_1,\ C_3 = C_3' - \mu V_1'.$$

Semi-functional key: Let $K_1', K_2', K_3', D', D_i'$ for all $i \neq j$ be secret key components normally generated by the **KeyGen** algorithm for user j. The semi-functional key generation algorithm will choose $\gamma \in \mathbb{Z}_p$ at random and modify the normal key as $K_3 = K_3'$, $D = D'$, $D_i = D_i'$ for all $i \neq j$ and

$$K_1 = K_1' - a\gamma P_2,\ K_2 = K_2' + \gamma P_2.$$

5 Conclusion

We have converted Waters dual-system IBE scheme from the setting of symmetric pairings to that of asymmetric pairings. This has been done in a systematic manner going through several stages of simplifications. We have described in details two IBE schemes (Scheme 1 and Scheme 6). Security of Scheme 1 is based on standard assumptions and reduces the sizes of ciphertexts and keys by 2 elements each from the original scheme of Waters. Scheme 6 is quite simple and minimal in the sense that both encryption and key generation use one randomiser each. The security of Scheme 6 is based on two standard assumptions and a natural and minimal extension of the DDH assumption for $\mathbb{G}_2$.

References

1. Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

2. 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.
3. 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.
4. Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.
5. Sanjit Chatterjee and Alfred Menezes. On cryptographic protocols employing asymmetric pairings – the role of ψ revisited. *Discrete Applied Mathematics*, 159(13):1311–1322, 2011.
6. Sanjit Chatterjee and Palash Sarkar. Trading Time for Space: Towards an Efficient IBE Scheme with Short(er) Public Parameters in the Standard Model. In Dong Ho Won and Seungjoo Kim, editors, *ICISC*, volume 3935 of *Lecture Notes in Computer Science*, pages 424–440. Springer, 2005.
7. Clifford Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In Bahram Honary, editor, *IMA Int. Conf.*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, 2001.
8. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.
9. 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.
10. Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2009.
11. Jeremy Horwitz and Ben Lynn. Toward Hierarchical Identity-Based Encryption. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer, 2002.
12. Allison Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. Cryptology ePrint Archive, Report 2011/490, 2011. `http://eprint.iacr.org/`.
13. Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer, 2010.
14. David Naccache. Secure and practical identity-based encryption. *IET Information Security*, 1(2):59–64, 2007.
15. J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27:701–717, October 1980.
16. 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.
17. Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 560–578. Springer, 2008.
18. Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997.
19. Nigel P. Smart and Frederik Vercauteren. On computable isomorphisms in efficient asymmetric pairing-based systems. *Discrete Applied Mathematics*, 155(4):538–547, 2007.
20. Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, 2005.
21. Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, 2009.

A Definitions and Security Models

We provide definitions and security models for identity-based encryption (IBE), hierarchical IBE (HIBE) and broadcast encryption (BE).

A.1 Identity-Based Encryption

An (IBE) scheme consists of four probabilistic algorithms – **Setup**, **KeyGen**, **Encrypt** and **Decrypt** – all of which run in time which is upper bounded by a polynomial in a security parameter κ.

1. **Setup** outputs the public parameters $\mathcal{PP}$ and the master secret key $\mathcal{MSK}$. $\mathcal{PP}$ is made public and $\mathcal{MSK}$ is kept secret.
2. **KeyGen** takes as input an identity id and $\mathcal{MSK}$ and returns a secret key $\mathcal{SK}_{\mathsf{id}}$ corresponding to id.
3. **Encrypt** takes as input a message M, an identity id, the public parameters $\mathcal{PP}$ and produces as output a ciphertext C.
4. **Decrypt** takes as input a ciphertext C, an identity id, the corresponding secret key $\mathcal{SK}_{\mathsf{id}}$, the public parameters $\mathcal{PP}$ and returns either the corresponding message M or returns $\perp$ indicating failure.

Security of an IBE scheme is modeled using a game between an adversary $\mathcal{A}$ and a challenger. There are several stages of the game which are described as follows.

Setup: The challenger takes as input a security parameter κ and runs the **Setup** algorithm of the IBE and provides the public parameters to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ makes a number of key extraction queries adaptively. It provides an id and the challenger computes the secret key corresponding to id and returns the key to $\mathcal{A}$.

Challenge: $\mathcal{A}$ provides a challenge identity id^* and two equal length messages M_0 and M_1 to the challenger with the restriction that id^* should not have been queried in **Phase 1**. The challenger then chooses a bit β uniformly at random from $\{0,1\}$ and returns an encryption of M_β to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ issues more key extraction queries like in **Phase 1** but it is not allowed to request a key for id^*.

Guess: $\mathcal{A}$ outputs a bit β'.

$\mathcal{A}$ wins the above game if $\beta = \beta'$. The advantage of $\mathcal{A}$ in breaking the security of the IBE scheme is defined in terms of the probability of the event that $\beta = \beta'$ in the above game as shown below.

$$\mathsf{Adv}_{\mathrm{IBE}}^{\mathcal{A}} = \left|\Pr[\beta = \beta'] - \frac{1}{2}\right|$$

The IBE scheme is said to be secure if a probabilistic adversary running in time bounded by a polynomial in the security parameter κ has a negligible (in κ) advantage of winning the above game.

A.2 Hierarchical Identity-Based Encryption

A HIBE scheme is defined by five probabilistic algorithms – **Setup**, **Encrypt**, **KeyGen**, **Delegate** and **Decrypt** . The runtime of all these algorithms is upper bounded by a polynomial in the security parameter κ.

1. **Setup** outputs the public parameters $\mathcal{PP}$ and the master secret $\mathcal{MSK}$. $\mathcal{PP}$ is made public and $\mathcal{MSK}$ is kept secret.
2. **KeyGen** takes as input an identity vector $\overrightarrow{\mathsf{id}}$ and master secret $\mathcal{MSK}$ and outputs the secret key $\mathcal{SK}_{\overrightarrow{\mathsf{id}}}$ corresponding to $\overrightarrow{\mathsf{id}}$.

3. **Encrypt** inputs a message M, an identity $\overrightarrow{\mathsf{id}}$, public parameters $\mathcal{PP}$ and returns a ciphertext $\mathcal{C}$.
4. **Delegate** inputs a depth ℓ identity vector $\overrightarrow{\mathsf{id}} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ along with a secret key $\mathcal{SK}_{\overrightarrow{\mathsf{id}}}$ and an identity $\mathsf{id}_{\ell+1}$. It outputs a secret key for the depth $d+1$ identity vector $(\mathsf{id}_1, \ldots, \mathsf{id}_{\ell+1})$.
5. **Decrypt** takes as input a ciphertext $\mathcal{C}$, an identity vector $\overrightarrow{\mathsf{id}}$, secret key $\mathcal{SK}_{\overrightarrow{\mathsf{id}}}$, $\mathcal{PP}$ and returns either the corresponding message M or $\perp$ indicating failure.

The security model we follow is the one proposed by Shi and Waters [17]. The earlier security definitions for HIBE [11, 9] made no distinction as to how a key is generated (by a fresh call to the **KeyGen** algorithm or by delegation). These definitions were sufficient for most of the earlier schemes since the two methods produced identically distributed keys. However, for the HIBE that we describe this is not the case. We have to rely on the Shi-Waters model in order to be able to keep track of the delegation paths of keys. Note that Shi-Waters model is weaker than the standard security model for HIBE.

As usual, the security for a HIBE scheme is modeled as a game between an adversary $\mathcal{A}$ and a simulator. Following are the different phases of the security game.

Setup: The challenger runs the **Setup** algorithm of the HIBE and gives the public parameters to $\mathcal{A}$. It also initializes a set $S = \emptyset$ which denotes the set of secret keys it has created but not revealed.

Phase 1: $\mathcal{A}$ makes a number of queries of the following types adaptively.

– **Create** The adversary $\mathcal{A}$ provides an identity $\overrightarrow{\mathsf{id}}$ for which the challenger creates a key $\mathcal{SK}_{\overrightarrow{\mathsf{id}}}$ but does not give it to $\mathcal{A}$. The challenger adds the secret key to S.
– **Delegate** $\mathcal{A}$ specifies a secret key $\mathcal{SK}_{\overrightarrow{\mathsf{id}}}$ in S and provides an identity id to the challenger. The challenger runs the delegation algorithm of the HIBE with inputs $\mathcal{PP}, \mathcal{SK}_{\overrightarrow{\mathsf{id}}}, \mathsf{id}$ and adds the resulting key for $(\overrightarrow{\mathsf{id}}, \mathsf{id})$ to S.
– **Reveal** $\mathcal{A}$ specifies an element $\mathcal{SK}$ of S. The challenger removes $\mathcal{SK}$ from S and returns it to $\mathcal{A}$.

Challenge: $\mathcal{A}$ provides a challenge identity $\overrightarrow{\mathsf{id}}^*$ and two equal length messages M_0 and M_1 to the challenger with the restriction that $\overrightarrow{\mathsf{id}}^*$ should not have been queried in **Phase 1**. The challenger then chooses a bit β uniformly at random from $\{0, 1\}$ and returns an encryption of M_β for $\overrightarrow{\mathsf{id}}^*$ to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ issues more key extraction queries as in **Phase 1** with the restriction that any revealed identity $\overrightarrow{\mathsf{id}}$ is not a prefix of $\overrightarrow{\mathsf{id}}^*$.

Guess: $\mathcal{A}$ outputs a bit β'.

$\mathcal{A}$ wins the above game if $\beta = \beta'$. The advantage of $\mathcal{A}$ in breaking the security of the IBE scheme is defined in terms of the probability of the event that $\beta = \beta'$ in the above game as shown below.

$$\mathsf{Adv}^{\mathcal{A}}_{\mathrm{HIBE}} = \left| \Pr[\beta = \beta'] - \frac{1}{2} \right|.$$

The HIBE scheme is said to be secure if all probabilistic adversaries running in time bounded by a polynomial have negligible advantage in winning the above game.

A.3 Broadcast Encryption

A broadcast encryption scheme is defined by four probabilistic algorithms whose runtimes are bounded above by a polynomial in the security parameter κ. The first three are executed by the broadcasting entity and the last one by a recipient of the broadcast message.

1. **Setup** takes as input the number of users n. It outputs a public/secret key pair $(\mathcal{PK}, \mathcal{SK})$.
2. **KeyGen** takes as input $\mathcal{SK}$, a user index $i \in \{1, \ldots, n\}$ and returns a private key $\mathcal{SK}_i$ for user i.
3. **Encrypt** inputs a subset $S \subseteq \{1, \ldots, n\}$ of users to whom the message is to be broadcast, the public key $\mathcal{PK}$ and the message M. It outputs a ciphertext $\mathcal{C}$.
4. **Decrypt** takes as input a ciphertext $\mathcal{C}$, a set S of users to whom $\mathcal{C}$ is encrypted and the user's secret key $\mathcal{SK}_i$. If $i \in S$ then the corresponding message M is returned.

Adaptive CPA security of a broadcast encryption (BE) scheme is defined via the following game between an adversary $\mathcal{A}$ and a challenger.

Setup: The challenger runs the **Setup** algorithm and gives the public key $\mathcal{PK}$ to $\mathcal{A}$.

Private Key Queries: $\mathcal{A}$ adaptively issues private key queries for a number of users from $\{1, \ldots, n\}$.

Challenge: $\mathcal{A}$ specifies two messages M_0, M_1 and a challenge set S^* such that, for every private key query i, $i \notin S^*$. The challenger chooses $\beta \in_R \{0, 1\}$ and returns to $\mathcal{A}$ the encryption of M_β for the set S^*.

Guess: The adversary outputs its guess β' for β.

The adversary wins the above game if $\beta = \beta'$. The advantage of $\mathcal{A}$ in breaking the security of the BE scheme is defined in terms of the probability of the event that $\beta = \beta'$ in the above game as shown below.

$$\mathsf{Adv}_{\mathrm{BE}}^{\mathcal{A}} = \left| \Pr[\beta = \beta'] - \frac{1}{2} \right| .$$

A BE scheme is said to be secure if for all probabilistic adversaries running in time bounded above by a polynomial in the security parameter κ, the advantage of winning the above game is negligible.

B Hardness Assumptions

Decision Diffie-Hellman (DDH) assumption. Let P_1 and P_2 be random generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. The DDH problem in $\mathbb{G}_1$ (denoted DDH1) is to decide, given $(P_1, x_1P_1, x_2P_1, P_2, Z_1)$, whether $Z_1 = x_1x_2P_1$ or Z_1 is a random element of $\mathbb{G}_1$. Here $x_1, x_2 \in_R \mathbb{Z}_p$.

Let $\mathcal{A}$ be a probabilistic polynomial time (PPT) algorithm that outputs either 0 or 1. Define its advantage in solving the DDH1 as follows.

$$\mathsf{Adv}_{\mathrm{DDH1}}^{\mathcal{A}} = |\Pr[\mathcal{A}(P_1, x_1P_1, x_2P_1, P_2, x_1x_2P_1) = 1] - \Pr[\mathcal{A}(P_1, x_1P_1, x_2P_1, P_2, Y_1) = 1]|$$

where $Y_1 \in_R \mathbb{G}_1$. The DDH1 assumption is that for every PPT algorithm $\mathcal{A}$, $\mathsf{Adv}_{\mathrm{DDH}}^{\mathcal{A}}$ is negligible for a suitable choice of parameters used to generate the groups. Similarly one can define the DDH assumption in $\mathbb{G}_2$ (DDH2).

Assumption DDH2v. Let P_1,P_2 be random generators of $\mathbb{G}_1$,$\mathbb{G}_2$ respectively and let $x_1, x_2, d, z \in_R \mathbb{Z}_p$. The problem is to decide, given $(P_1, dP_1, dzP_1, zx_1P_1, P_2, dP_2, x_1P_2, x_2P_2, Z_2)$, whether $Z_2 = x_1x_2P_2$ or Z_2 is a random element of $\mathbb{G}_2$.

Let $\mathcal{A}$ be a probabilistic polynomial time (PPT) algorithm that outputs a bit. Define its advantage in solving breaking DDH2v as follows.

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}}_{\text{DDH2v}} = |\Pr[&\mathcal{A}(P_1, dP_1, dzP_1, zx_1P_1, P_2, dP_2, x_1P_2, x_2P_2, x_1x_2P_2) = 1] \\ &- \Pr[\mathcal{A}(P_1, dP_1, dzP_1, zx_1P_1, P_2, dP_2, x_1P_2, x_2P_2, Y_2) = 1]|\end{aligned}$$

where $Y_1 \in_R \mathbb{G}_2$. The assumption is that $\mathsf{Adv}^{\mathcal{A}}_{\text{DDH2v}}$ is negligible in the parameters that define the size of the groups. In Section B.1, we show the hardness of the DDH2v assumption in the generic group model.

Decision Linear (DLin) assumption. Let P_1, F_1, H_1 be random generators of $\mathbb{G}_1$ and P_2, F_2, H_2 be random generators of $\mathbb{G}_2$. Let $x_1, x_2 \in_R \mathbb{Z}_p$ and $Y_2 \in_R \mathbb{G}_2$. Let $\mathcal{A}$ be a PPT algorithm that outputs either 0 or 1. Define its advantage in solving the DLin problem as follows.

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}}_{\text{DLin}} =| \Pr[&\mathcal{A}(P_1, F_1, H_1, P_2, F_2, H_2, x_1P_2, x_2F_2, (x_1 + x_2)H_2) = 1] \\ &- \Pr[\mathcal{A}(P_1, F_1, H_1, P_2, F_2, H_2, x_1P_2, x_2F_2, Y_2) = 1]|\end{aligned}$$

The DLin assumption is that for every PPT algorithm $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}_{\text{DLin}}$ is negligible.

Decisional Bilinear Diffie-Hellman (DBDH) assumption. Let P_1, P_2 be random generators of $\mathbb{G}_1$,$\mathbb{G}_2$ respectively and let x_1, x_2, x_3 be random elements of $\mathbb{Z}_p$. Let Y_T be a random element of $\mathbb{G}_T$. For a probabilistic polynomial time (PPT) algorithm $\mathcal{A}$ that outputs either 0 or 1, define its advantage in solving the DBDH problem as follows.

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}}_{\text{DBDH}} =| \Pr[&\mathcal{A}(P_1, x_1P_1, x_2P_1, x_3P_1, P_2, x_1P_2, x_2P_2, x_3P_2, e(P_1, P_2)^{x_1x_2x_3}) = 1] \\ &- \Pr[\mathcal{A}(P_1, x_1P_1, x_2P_1, x_3P_1, P_2, x_1P_2, x_2P_2, x_3P_2, Y_T) = 1]|\end{aligned}$$

The DBDH assumption is that for every PPT algorithm $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}_{\text{DBDH}}$ is negligible. Informally, it should be computationally hard to distinguish between $e(P_1, P_2)^{x_1x_2x_3}$ and a random element of $\mathbb{G}_T$.

B.1 Generic Security of DDH2v

Here we will provide a proof of the security of assumption DDH2v in the generic group model. The generic group model is an idealised model introduced by [18] in which lower bounds on computational complexity of solving certain problems can be obtained without looking into the structure of the actual groups that are used in a protocol. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an asymmetric bilinear group where no isomorphisms from $\mathbb{G}_1$ to $\mathbb{G}_2$ or $\mathbb{G}_2$ to $\mathbb{G}_1$ are efficiently computable. The elements of groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are encoded as uniform random strings so that the adversary can only test for equality of group elements. Four oracles are provided to the adversary out of which three simulate the group actions in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ and the fourth one simulates the bilinear map e. The group encodings are modeled as three injective maps $\sigma_1 : \mathbb{Z}_p \to \Sigma_1$, $\sigma_2 : \mathbb{Z}_p \to \Sigma_2$ and $\sigma_T : \mathbb{Z}_p \to \Sigma_T$ where $\mathbb{E}_1, \mathbb{E}_2, \mathbb{E}_T \subset \{0,1\}^*$. The following theorem provides an upper bound on the advantage of an adversary that solves the DDH2v problem in a generic bilinear group.

Theorem 1. *Let $\mathcal{A}$ be an algorithm that attempts to solve the* DDH2v *problem in the generic group model making at most m queries to the oracles computing the group actions in $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ and the bilinear map e. If $d, z, x_1, x_2, y \in_R \mathbb{Z}_p$, $b \in_R \{0,1\}$ with $y_b = x_1x_2$ and $y_{1-b} = y$ and $\sigma_1, \sigma_2, \sigma_T$ are random encodings, then given $p, \sigma_1(1), \sigma_1(d), \sigma_1(dz), \sigma_1(zx_1), \sigma_2(1), \sigma_2(d), \sigma_2(x_1), \sigma_2(x_2), \sigma_2(y_0), \sigma_2(y_1)$ the advantage ε of $\mathcal{A}$ in solving the problem is bounded above by*

$$\varepsilon \le \frac{3(m+10)^2}{2p}.$$

Proof. Let $\mathcal{B}$ denote an algorithm that simulates the generic bilinear group for $\mathcal{A}$. $\mathcal{B}$ maintains three lists

$$L_1 = \{(F_{1,i}, \sigma_{1,i}) : i = 0, 1, \dots, \delta_1 - 1\},$$
$$L_2 = \{(F_{2,i}, \sigma_{2,i}) : i = 0, 1, \dots, \delta_2 - 1\},$$
$$L_T = \{(F_{T,i}, \sigma_{T,i}) : i = 0, 1, \dots, \delta_T - 1\}$$

such that at each step δ of the game the relation $\delta_1 + \delta_2 + \delta_T = \delta + 10$ holds. Here $F_{\star,\star}$'s are multivariate polynomials over 6 variables d, z, x_1, x_2, y_0, y_1 and $\sigma_{\star,\star}$'s are strings from $\{0,1\}^*$. At the beginning of the game i.e., $\delta = 0$, the lists are intialized by setting $\delta_1 = 4$, $\delta_2 = 6$ and $\delta_T = 0$. The polynomials $1, d, dz, zx_1$ are assigned to $F_{1,0}, F_{1,1}, F_{1,2}, F_{1,3}$ and $1, d, x_1, x_2, y_0, y_1$ to $F_{2,0}, F_{2,1}, F_{2,2}, F_{2,3}, F_{2,4}, F_{2,5}$ respectively. The encodings for these polynomials are strings uniformly chosen from $\{0,1\}^*$ without repetition. We assume that $\mathcal{A}$ queries the oracles on strings previously obtained from $\mathcal{B}$ and $\mathcal{B}$ can easily obtain the index of a given string $\sigma_{j,i}$ in the list L_j. The oracles are simulated as follows.

Group actions in $\mathbb{G}_1$,$\mathbb{G}_2$ and $\mathbb{G}_T$: Consider the group $\mathbb{G}_1$. $\mathcal{A}$ submits two strings $\sigma_{1,i}$ and $\sigma_{1,j}$ and a selection bit indicating addition or subtraction. $\mathcal{B}$ first computes $F_{1,\delta_1} = F_{1,i} \pm F_{1,j}$. If there exists an index k with $0 \le k < \delta_1$ such that $F_{1,\delta_1} = F_{1,k}$ then $\mathcal{B}$ sets $\sigma_{1,\delta_1} = \sigma_{1,k}$. Otherwise it sets σ_{1,δ_1} to a uniform random string from $\{0,1\}^* \setminus \{\sigma_{1,0}, \dots, \sigma_{1,\delta_1-1}\}$. $\mathcal{B}$ then adds the pair $(F_{1,\delta_1}, \sigma_{1,\delta_1})$ to L_1, returns σ_{1,δ_1} to $\mathcal{A}$ and increments δ_1 by one.

Group actions in $\mathbb{G}_2$ and $\mathbb{G}_T$ are simulated similarly with the selection bit indicating multiplication or division in case of $\mathbb{G}_T$.

Bilinear map: $\mathcal{A}$ submits two operands $\sigma_{1,i}, \sigma_{2,j}$. $\mathcal{B}$ fetches the corresponding polynomials $F_{1,i}, F_{2,j}$ and computes $F_{T,\delta_T} = F_{1,i} \cdot F_{2,j}$. If for some k with $0 \le k < \delta_T$, $F_{T,\delta_T} = F_{T,k}$ then set $\sigma_{T,\delta_T} = \sigma_{T,k}$; otherwise σ_{T,δ_T} is set to a random string chosen uniformly from $\{0,1\}^* \setminus \{\sigma_{T,0}, \dots, \sigma_{T,\delta_T-1}\}$. $\mathcal{B}$ then adds the pair $(F_{T,\delta_T}, \sigma_{T,\delta_1})$ to L_1, returns σ_{T,δ_T} to $\mathcal{A}$ and increments δ_T by one.

$\mathcal{A}$ makes at most m oracle queries, terminates and returns a bit b' to the simulator. Let $\boldsymbol{v} = (d, z, x_1, x_2.y_0, y_1)$ denote the vector consisting of variables over which the polynomials are defined. Now the simulator chooses at random $d^*, z^*, {x_1}^*, {x_2}^*, y^* \in \mathbb{Z}_p$ and $b \in \{0,1\}$ and sets ${y_b}^* = {x_1}^*{x_2}^*$, ${y_{1-b}}^* = y^*$. Let $\boldsymbol{v}^* = (d^*, z^*, {x_1}^*, {x_2}^*, {y_0}^*, {y_1}^*)$. $\mathcal{B}$ assigns $\boldsymbol{v}^*$ to the variables $\boldsymbol{v}$. The simulation provided by $\mathcal{B}$ is perfect unless this assignment causes any of the following to hold.

1. $F_{1,i} - F_{1,j} = 0$ for some $i \neq j$ and $F_{1,i} \neq F_{1,j}$.
2. $F_{2,i} - F_{2,j} = 0$ for some $i \neq j$ and $F_{2,i} \neq F_{2,j}$.
3. $F_{T,i} - F_{T,j} = 0$ for some $i \neq j$ and $F_{T,i} \neq F_{T,j}$.

Let F denote the event that atleast one of the above holds, indicating failure. The following result by Schwartz [15] will be used in arguing that the event failure occurs with low probability. Let p be a prime number and $F(Z_1, \ldots, Z_k)$ be a non-zero polynomial in $\mathbb{Z}_p[Z_1, \ldots, Z_k]$ of degree d. Then, if $z_1, \ldots, z_k$ are uniform elements of $\mathbb{Z}_p$, the probability that $F(z_1, \ldots, z_k) = 0$ is at most d/p.

The simulation is perfect when F does not occur and in such a case the bit b is information theoretically hidden from the adversary. To see this, observe that all variables except y_b and y_{1-b} are independent of the bit b. Since y_b is x_1x_2 which is a polynomial of degree 2, the adversary will win it somehow produces x_1x_2 using combinations of polynomials from L_1 and L_2. The only degree two polynomials that can be constructed are $d^2, dx_1, dx_2, dz, zx_1$ or a sum of these. $\mathcal{A}$ could also try to engineer a degree 3 polynomial in L_T composed of x_1x_2. The only such polynomial is zx_1x_2 which can be constructed using $\sigma_1(zx_1)$ and $\sigma_2(x_2)$. However, to find out the bit b, $\mathcal{A}$ will need $\sigma_1(z)$ which is not available in the instance. So $\Pr[b = b'|\neg\mathsf{F}] = 1/2$.

We now only need to obtain a bound on the probability that F occurs. For fixed i and j, $F_{1,i} - F_{1,j}$ is a polynomial of degree at most 2 and hence is zero at a random $\boldsymbol{v}^*$ with probability at most $2/p$. Similary $F_{2,i} - F_{2,j}$ vanishes at $\boldsymbol{v}^*$ with probability at most $1/p$. The list L_3 consists of polynomials of degree at most 3 which implies that the third case holds with probability at most $3/p$. There are totally $\binom{\delta_1}{2}$, $\binom{\delta_2}{2}$, $\binom{\delta_T}{2}$ pairs of polynomials from L_1,L_2, L_T respectively. Also since there are at most m queries we have $\delta_1 + \delta_2 + \delta_T = \delta + 10 \leq m + 10$. It now follows that

$$\begin{aligned}\Pr[\mathsf{F}] &\leq \binom{\delta_1}{2}\frac{2}{p} + \binom{\delta_2}{2}\frac{1}{p} + \binom{\delta_T}{2}\frac{3}{p} \\ &\leq \frac{3(m+10)^2}{p}.\end{aligned}$$

We have

$$\begin{aligned}\Pr[b = b'] &= \Pr[b = b'|\neg\mathsf{F}]\Pr[\neg\mathsf{F}] + \Pr[b = b'|\mathsf{F}]\Pr[\mathsf{F}] \\ &\leq \Pr[b = b'|\neg\mathsf{F}](1 - \Pr[\mathsf{F}]) + \Pr[\mathsf{F}] \\ &\leq \frac{1}{2} + \frac{1}{2}\Pr[\mathsf{F}]\end{aligned}$$

and

$$\Pr[b = b'] \geq \Pr[b = b'|\neg\mathsf{F}](1 - \Pr[\mathsf{F}]) = \frac{1}{2} - \frac{1}{2}\Pr[\mathsf{F}]$$

together resulting in the required bound on the advantage as follows.

$$\left|\Pr[b = b'] - \frac{1}{2}\right| \leq \frac{\Pr[\mathsf{F}]}{2} \leq \frac{3(m+10)^2}{2p}.$$

□

C Lewko's IBE Scheme Using Asymmetric Pairing

In a recent work [12], Lewko has systematically described a method for converting a dual-system IBE scheme from the setting of composite order groups to that of prime order groups. For simplicity, the description in [12] has been given in terms of symmetric pairing. It is fairly easy to convert it to the setting of asymmetric pairing. In this section, we briefly describe this scheme.

For the convenience of presentation, some notation has been introduced in [12]. For $\overrightarrow{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_p^n$ and an elliptic curve point P, define

$$\overrightarrow{v}P \stackrel{\Delta}{=} (v_1P, \ldots, v_nP).$$

Note that computing $\overrightarrow{v}P$ requires n scalar multiplications. This notation is extended as follows.

$$\begin{aligned} a\overrightarrow{v}P &\stackrel{\Delta}{=} (av_1P, \ldots, av_nP); \\ (\overrightarrow{v} + \overrightarrow{w})P &\stackrel{\Delta}{=} ((v_1 + w_1)P, \ldots, (v_n + w_n)P); \\ e_n(\overrightarrow{v}P_1, \overrightarrow{w}P_2) &\stackrel{\Delta}{=} \prod_{i=1}^{n} e(v_iP_1, w_iP_2) = e(P_1, P_2)^{\overrightarrow{v}\cdot\overrightarrow{w}}. \end{aligned}$$

Here $\overrightarrow{v} \cdot \overrightarrow{w}$ denotes the dot product. The computation of $e_n(\cdot, \cdot)$ requires n pairings.

Let $\mathbb{D} = (\overrightarrow{d_1}, \ldots, \overrightarrow{d_6})$ be a basis for $\mathbb{Z}_p^6$ and $\mathbb{D}^* = (\overrightarrow{d_1^*}, \ldots, \overrightarrow{d_6^*})$ be another basis of $\mathbb{Z}_p^6$ such that $\overrightarrow{d_i} \cdot \overrightarrow{d_j^*} \equiv 0 \bmod p$ for $i \neq j$; and $\overrightarrow{d_i} \cdot \overrightarrow{d_i^*} = \psi$ for all i, where ψ is a uniform random element of $\mathbb{Z}$. Such a pair of bases is called "orthonormal" in [12].

An asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ will be used. Elements P_1 and P_2 are respectively generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. To set up the scheme in [12], first one chooses a random pair of orthonormal bases $\mathbb{D}$ and $\mathbb{D}^*$. Further, choose random scalars α, θ, σ from $\mathbb{Z}_p$. The first four vectors of $\mathbb{D}$ and $\mathbb{D}^*$ are actually used in the scheme. The other two elements of the bases are used to define the semi-functional spaces.

$\mathcal{PP}$: $e(P_1, P_2)^{\alpha\theta\overrightarrow{d_1}\cdot\overrightarrow{d_1^*}}$, $\overrightarrow{d_1}P_1$, $\overrightarrow{d_2}P_1$, $\overrightarrow{d_3}P_1$, $\overrightarrow{d_4}P_1$.

$\mathcal{MSK}$: $\theta\overrightarrow{d_1^*}P_2$, $\alpha\theta\overrightarrow{d_2^*}P_2$, $\theta\overrightarrow{d_2^*}P_2$, $\sigma\overrightarrow{d_3^*}P_2$, $\sigma\overrightarrow{d_4^*}P_2$.

Note that the $\mathcal{PP}$ consists of 24 elements of $\mathbb{G}_1$ and the $\mathcal{MSK}$ consists of 30 elements of $\mathbb{G}_2$. The public key generator may also keep the scalars α, θ, σ and the bases $\mathbb{D}$ and $\mathbb{D}^*$ as the secret. This will lead to a faster key generation algorithm.

Key generation: The input is an identity id. Choose random elements r_1 and r_2 from $\mathbb{Z}_p$. The secret key for the identity id is $\mathcal{SK}_{\mathsf{id}}$ and is defined to be the following 6-tuple of elements of $\mathbb{G}_2$.

$$\mathcal{SK}_{\mathsf{id}} = \left((\alpha + r_1\mathsf{id})\theta\overrightarrow{d_1^*} - r_1\theta\overrightarrow{d_2^*} + r_2\mathsf{id}\sigma\overrightarrow{d_3^*} - r_2\sigma\overrightarrow{d_4^*}\right) P_2.$$

A key consists of 6 elements of $\mathbb{G}_2$. Generation of $\mathcal{SK}_{\mathsf{id}}$ from $\mathcal{MSK}$ requires 30 scalar multiplications in $\mathbb{G}_2$. But, generation from the scalars α, θ, σ and the bases $\mathbb{D}$ and $\mathbb{D}^*$ will require 6 scalar multiplications in $\mathbb{G}_2$.

Encrypt: The input consists of an identity id and a message $M \in \mathbb{G}_T$. Choose random elements s_1 and s_2 from $\mathbb{Z}_p$. The ciphertext is defined to be (C_1, C_2) where C_1 and C_2 are elements of $\mathbb{G}_1$ and are defined as follows.

$$C_1 = M \times \left(e(P_1, P_2)^{\alpha\theta\overrightarrow{d_1}\cdot\overrightarrow{d_1^*}}\right)^{s_1};$$
$$C_2 = \left(s_1\overrightarrow{d_1} + s_1\mathsf{id}\overrightarrow{d_2} + s_2\mathsf{id}\overrightarrow{d_3} + s_2\mathsf{id}\overrightarrow{d_4}\right)P_1.$$

A ciphertext consists of 6 elements of $\mathbb{G}_1$. Generation of a ciphertext from the public parameters requires 24 scalar multiplications in $\mathbb{G}_1$, i.e., the four 6-tuples $s_1(\overrightarrow{d_1}P_1)$, $s_1\mathsf{id}(\overrightarrow{d_2}P_1)$, $s_2\mathsf{id}(\overrightarrow{d_3})P_1$ and $s_2\mathsf{id}(\overrightarrow{d_4}P_1)$ have to generated and added together. Generation of each of these 6-tuples will require 6 scalar multiplications in $\mathbb{G}_1$ for a total of 24 scalar multiplications.

Decrypt: Given $\mathcal{PP}$, an identity id, a corresponding key $\mathcal{SK}_{\mathsf{id}}$ and a ciphertext (C_1, C_2), decryption can be compactly described as follows: return $C_1/e_6(C_2, \mathcal{SK}_{\mathsf{id}})$. This requires 6 pairing operations.

Descriptions of the semi-functional keys and ciphertexts are given in [12] in terms of symmetric pairing. These can also be easily converted to the setting of asymmetric pairings. The security of the above scheme should also be based on DLin assumptions in $\mathbb{G}_1$ and $\mathbb{G}_2$.

Further simplications to the above scheme may be possible along the lines of the simplifications we made to Waters' 2009 scheme. Then the hardness assumption is unlikely to be DLin or SXDH; possibly a combination of DDH1, DDH2v and DBDH may work. On the other hand, the amount of reduction in parameters and improvement in efficiency is not clear. In particular, it is unlikely that the resulting scheme would be better than Scheme 6.

D Scheme 1

In this section, we provide the description and the security argument for Scheme 1.

D.1 Description of Scheme 1

Choose generators $P_1 \in_R \mathbb{G}_1$ and $P_2 \in_R \mathbb{G}_2$; random elements $Q_1, W_1, U_1 \in \mathbb{G}_1$ and $Q_2, W_2, U_2 \in \mathbb{G}_2$ such that $Q_2 \sim Q_1$, $U_2 \sim U_1$ and $W_2 \sim W_1$. Let v, v' be chosen randomly from $\mathbb{Z}_p$ and set $V_2 = vP_2$, $V_2' = v'P_2$. Pick α, a, b at random from $\mathbb{Z}_p$. Set $\tau = v + av'$ so that $\tau P_2 = V_2 + aV_2'$.

$\mathcal{PP}$: $(P_1, aP_1, bP_1, abP_1, \tau P_1, b\tau P_1, Q_1, W_1, U_1, e(P_1, P_2)^{b\alpha})$.
$\mathcal{MSK}$: $(P_2, \alpha P_2, bP_2, V_2, V_2', Q_2, U_2, W_2)$.

Encrypt$(M, \mathsf{id}, \mathcal{PP})$: Choose random s, t, ctag from $\mathbb{Z}_p$. $\mathcal{C}$ is $(C_0, C_1, \ldots, C_5, E_1, E_2, \mathsf{ctag})$ where the elements are computed as follows.

$C_0 = M \cdot e(P_1, P_2)^{b\alpha s}$,
$C_1 = bsP_1,\ C_2 = basP_1,\ C_3 = asP_1,\ C_4 = -\tau sP_1,\ C_5 = -\tau bsP_1 + tW_1$,
$E_1 = t(\mathsf{id}Q_1 + \mathsf{ctag}W_1 + U_1),\ E_2 = tP_1$.

KeyGen$(\mathsf{id}, \mathcal{MSK}, \mathcal{PP})$: Choose random $r_1, r_2, z, \mathsf{ktag}$ from $\mathbb{Z}_p$ and let $r = r_1 + r_2$. $\mathcal{SK}_{\mathsf{id}}$ is $(K_1, \ldots, K_5, D, \mathsf{ktag})$ where the elements are computed as follows.

$$K_1 = \alpha P_2 + rV_2,\ K_2 = rV_2' - zP_2,\ K_3 = bzP_2,\ K_4 = br_2P_2,\ K_5 = r_1P_2,$$
$$D = r_1(\mathsf{id}Q_2 + \mathsf{ktag}W_2 + U_2).$$

Decrypt $(\mathcal{C}, \mathsf{id}, \mathcal{SK}_{\mathsf{id}}, \mathcal{PP})$: Decryption succeeds only when $\mathsf{ctag} \neq \mathsf{ktag}$, an event which occurs with overwhelming probability. Decryption is done in several stages. Using the relations $r = r_1 + r_2$ and $V_2 + aV_2' = \tau P_2$, compute

$$\begin{aligned} A_1 &= e(C_1, K_1)e(C_2, K_2)e(C_3, K_3)e(C_4, K_4)e(C_5, K_5) \\ &= e(P_1, P_2)^{b\alpha s} e(P_1, V_2)^{bsr} e(P_1, V_2')^{absr} e(P_1, P_2)^{-basz} e(P_1, P_2)^{basz} \\ &\times e(P_1, P_2)^{-\tau bsr_2} e(P_1, P_2)^{-\tau bsr_1} e(W_1, P_2)^{tr_1} \\ &= e(P_1, P_2)^{b\alpha s} e(P_1, V_2 + aV_2' - \tau P_2)^{bsr} e(W_1, P_2)^{tr_1} \\ &= e(P_1, P_2)^{b\alpha s} e(P_1, W_2)^{tr_1} \end{aligned}$$

The last step follows from the fact that the discrete log of W_1 to base P_1 is same as that of W_2 to base P_2. Then if $\mathsf{ctag} \neq \mathsf{ktag}$ compute

$$\begin{aligned} A_2 &= \left(\frac{e(E_1, K_5)}{e(E_2, D)}\right)^{1/(\mathsf{ctag}-\mathsf{ktag})} \\ &= \left(\frac{e(t(\mathsf{id}Q_1 + \mathsf{ctag}W_1 + U_1), r_1P_2)}{e(tP_1, r_1(\mathsf{id}Q_2 + \mathsf{ktag}W_2 + U_2))}\right)^{1/(\mathsf{ctag}-\mathsf{ktag})} \\ &= \left(\frac{e(W_1, P_2)^{tr_1\mathsf{ctag}}}{e(P_1, W_2)^{tr_1\mathsf{ktag}}}\right)^{1/(\mathsf{ctag}-\mathsf{ktag})} \\ &= e(P_1, W_2)^{tr_1} \end{aligned}$$

Unmask C_0 to get the message M as $M = (C_0 \cdot A_2)/A_1$.

We now define semi-functional ciphertexts and keys. They are used only in the security reduction and they cannot be computed without knowledge of the secret elements.

Semi-functional ciphertext: Let $C_0', C_1', C_2', C_3', C_4', C_5', E_1', E_2', \mathsf{ctag}$ be ciphertext elements normally generated by the **Encrypt** algorithm for message M and identity id. Let V_1' be an element of $\mathbb{G}_1$ such that $V_1' \sim V_2'$. Choose $\mu \in \mathbb{Z}_p$ at random. The semi-functional ciphertext generation algorithm will modify the normal ciphertext as: $C_0 = C_0'$, $C_1 = C_1'$, $E_1 = E_1'$, $E_2 = E_2'$ and

$$C_2 = C_2' + b\mu P_1,\ C_3 = C_3' + \mu P_1,\ C_4 = C_4' - \mu V_1',\ C_5 = C_5' - b\mu V_1'.$$

Semi-functional key: Let $K_1', K_2', K_3', K_4', K_5', D', \mathsf{ktag}$ be secret key components normally generated by the **KeyGen** algorithm for identity id. Choose at random $\gamma \in \mathbb{Z}_p$. The semi-functional key generation algorithm will modify the normal key as: $K_3 = K_3'$, $K_4 = K_4'$, $K_5 = K_5'$, $D = D'$ and

$$K_1 = K_1' - a\gamma P_2,\ K_2 = K_2' + \gamma P_2.$$

One can decrypt a semi-functional ciphertext with a normal key and a normal ciphertext with a semi-functional key. This is easily seen by verifying that

$$e(b\mu P_1, K_2)e(\mu P_1, K_3)e(-\mu V_1', K_4)e(-b\mu V_1', K_5) = 1_T$$

and $e(C_1, -a\gamma P_2)e(C_2, \gamma P_2) = 1_T$ where K_2, K_3, K_4, K_5 and C_1, C_2 are normal key and ciphertext components respectively. However, decryption of a semi-functional ciphertext with a semi-functional key will fail because the masking factor $e(P_1, P_2)^{b\alpha s}$ will be blinded by an additional factor of $e(P_1, P_2)^{b\mu\gamma}$.

D.2 Security Proof

As is usual, the proof goes through a sequence of games. Let Game$_{real}$ denote the real security game. Game$_0$ is just like Game$_{real}$ except that the challenge ciphertext is a semi-functional encryption of the chosen message. Let q be the number of key extraction queries made by the adversary during the attack. Define Game$_k$ for $1 \leq k \leq q$ such that the first k keys returned to the adversary are semi-functional and the rest are normal. Let Game$_{final}$ be defined similar to Game$_q$ except that now the challenge ciphertext is a semi-functional encryption of a random message. Let X_{real}, X_k and X_{final} denote the events that the adversary wins in Game$_{real}$, Game$_k$ and Game$_{final}$ for $0 \leq k \leq q$ respectively.

Lemma 1. *If there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{real}} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_0} = \varepsilon$, then we can build an algorithm $\mathcal{B}$ having advantage ε in solving the* DDH1 *problem.*

Proof. The algorithm $\mathcal{B}$ receives $(P_1, sP_1, aP_1, P_2, Z_1)$ as an instance of DDH1. We describe how it will simulate each phase in the security game.

Setup: $\mathcal{B}$ chooses random elements $\alpha, b, y_v, y_v', y_q, y_w, y_u$ from $\mathbb{Z}_p$ and sets the parameters as follows.

$$P_1 = P_1, sP_1 = sP_1, aP_1 = aP_1, Q_1 = y_q P_1, W_1 = y_w P_1, U_1 = y_u P_1,$$
$$P_2 = P_2, V_2 = y_v P_2, V_2' = y_v' P_2.$$

This implicitly sets $\tau = y_v + a y_v'$. Using this, the element τP_1 can be computed as $y_v P_1 + y_v'(aP_1)$. The simulator computes the remaining parameters using b, α and gives the following public parameters to $\mathcal{A}$.

$$\mathcal{PP} = \{\mathcal{G}, P_1, P_2, aP_1, bP_1, baP_1, \tau P_1, b\tau P_1, Q_1, W_1, U_1, e(P_1, P_2)^{b\alpha}\}$$

where $\mathcal{G}$ is the group description with groups $\mathbb{G}_1$ and $\mathbb{G}_2$ given by the generators P_1 and P_2. Note that a and s (randomiser for challenge ciphertext) come from the assumption and are not known to $\mathcal{B}$.

Phase 1: $\mathcal{A}$ makes a number of key extract queries. $\mathcal{B}$ knows the master secret and using that it returns a normal key generated using the **KeyGen** algorithm for every key extract query made by $\mathcal{A}$.

Challenge: $\mathcal{B}$ receives the target identity id^* and two messages M_0 and M_1 from $\mathcal{A}$. It chooses $\beta \in \{0, 1\}$ at random. To encrypt M_β, $\mathcal{B}$ chooses t, ctag^* at random from $\mathbb{Z}_p$ and computes the ciphertext elements as follows.

$$C_0 = M_\beta \cdot e(sP_1, P_2)^{b\alpha}$$

$$C_1 = b(sP_1),\ C_2 = bZ_1,\ C_3 = Z_1,\ C_4 = -y_v(sP_1) - y'_v Z_1,\ C_5 = -by_v(sP_1) - by'_v Z_1 + tW_1$$
$$E_1 = t(\mathsf{id}^* Q_1 + \mathsf{ctag}^* W_1 + U_1),\ E_2 = tP_1$$

$\mathcal{B}$ returns $\mathcal{C}^* = (C_0, C_1, C_2, C_3, C_4, C_5, E_1, E_2, \mathsf{ctag}^*)$ to $\mathcal{A}$. Now either $Z_1 = asP_1$ or $Z_1 = (as+c)P_1$ for some $c \in_R \mathbb{Z}_p$. In the first case we have

$$C_2 = basP_1,\ C_3 = asP_1$$

$$C_4 = -(y_v + ay'_v)sP_1 = -\tau sP_1$$
$$C_5 = -b(y_v + ay'_v)sP_1 + tW_1 = -b\tau P_1 + tW_1$$

making $\mathcal{C}^*$ a normal ciphertext. In the second case, we have

$$C_2 = basP_1 + bcP_1,\ C_3 = asP_1 + cP_1$$

$$C_4 = -(y_v + ay'_v)sP_1 - y'_v cP_1 = -\tau sP_1 - cV'_1$$
$$C_5 = -b(y_v + ay'_v)sP_1 - by'_v cP_1 + tW_1 = -b\tau sP_1 + tW_1 - bcV'_1$$

Then $\mathcal{C}^*$ is a semi-functional encryption of M_β with $\mu = c$. Note that, to check whether $\mathcal{C}^*$ is semi-functional or not, $\mathcal{B}$ itself could try to decrypt it with a semi-functional key for id^*. However since aP_2 is not known to $\mathcal{B}$, it cannot create such a key.

Phase 2: As in first phase, $\mathcal{B}$ returns a normal key for every query.

Guess: The adversary returns its guess β' to $\mathcal{B}$.

If $\mathcal{C}^*$ is normal then $\mathcal{B}$ simulates Game$_{real}$ and if it is semi-functional, $\mathcal{B}$ simulates Game$_0$. Therefore if $\mathcal{A}$ is able to distinguish between Game$_{real}$ and Game$_0$ i.e., if $\beta = \beta'$, then the simulator can decide whether $\mathcal{C}^*$ is normal or semi-functional and thus solve the DDH1 problem with advantage

$$\mathsf{Adv}^{\mathcal{B}}_{\text{DDH1}} = |\Pr[X_{real}] - \Pr[X_0]| = \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{real}} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_0} = \varepsilon$$

□

Lemma 2. *If there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_k} = \varepsilon$, then we can build an algorithm $\mathcal{B}$ having advantage ε in solving the* DLin *problem.*

Proof. The algorithm $\mathcal{B}$ receives $(P_1, F_1, H_1, P_2, F_2, H_2, x_1P_2, x_2F_2, Z_2)$ as an instance of the DLin problem.

Setup: $\mathcal{B}$ chooses random elements $\alpha, a, \lambda, \nu, y'_v, y_q, y_w, y_u$ from $\mathbb{Z}_p$ and sets the parameters as follows.

$$P_1 = P_1, bP_1 = F_1, P_2 = P_2, bP_2 = F_2, V_2 = -aH_2, V'_2 = H_2 + y'_v P_2$$
$$Q_1 = -\lambda F_1 + y_q P_1, U_1 = -\nu F_1 + y_u P_1, W_1 = F_1 + y_w P_1$$

This sets $\tau = ay'_v$ which is known to the simulator. The remaining parameters required to provide $\mathcal{PP}$ to $\mathcal{A}$ are computed using a, α and τ and other elements of the DLin instance as shown below.

$$abP_1 = aF_1,\ \tau P_1 = ay'_v P_1,\ b\tau P_1 = ay'_v F_1,\ e(P_1, P_2)^{b\alpha} = e(F_1, P_2)^\alpha$$

Phases 1 and 2: Let $\mathsf{id}_1, \mathsf{id}_2, \ldots, \mathsf{id}_q$ denote the identities for which $\mathcal{A}$ requests the corresponding secret keys. In Game$_k$, $\mathcal{B}$ changes the answer to k'th query from a normal key to a semi-functional one. Let $\mathcal{SK}'_{\mathsf{id}_i} = (K'_1, K'_2, K'_3, K'_4, K'_5, D', \mathsf{ktag})$ be a normally generated key for id_i. Note that $\mathcal{B}$ can create such a key because it knows the master secret. For $i < k$, the simulator should return a semi-functional key for id_i. $\mathcal{B}$ chooses $\gamma \in \mathbb{Z}_p$ at random, sets

$$K_1 = K'_1 - a\gamma P_2,\ K_2 = K'_2 + \gamma P_2,\ K_3 = K'_3,\ K_4 = K'_4,\ K_5 = K'_5,\ D = D'$$

and then returns the modified key $\mathcal{SK}_{\mathsf{id}_i}$ to $\mathcal{A}$. For $i > k$, $\mathcal{B}$ returns the normal key generated using the algorithm **KeyGen**.

When $i = k$, i.e., for identity id_k, suppose that a normal key $\mathcal{SK}'_{\mathsf{id}_k}$ is generated with r'_1, r'_2, z' as the randomisers and $\mathsf{ktag} = \lambda\mathsf{id}_k + \nu$. The simulator then modifies the key elements as follows.

$$K_1 = K'_1 - aZ_2,\ K_2 = K'_2 + Z_2 + y'_v(x_1 P_2),\ K_3 = K'_3 + y'_v(x_2 F_2)$$

$$K_4 = K'_4 + x_2 F_2,\ K_5 = K'_5 + x_1 P_2,\ D = D' + (y_q \mathsf{id}_k + y_w \mathsf{ktag} + y_u)(x_1 P_2)$$

implicitly setting $r_1 = r'_1 + x_1$, $r_2 = r'_2 + x_2$, $r = r' + x_1 + x_2$ and $z = z' + y'_v x_2$. Now $\mathcal{B}$ returns the secret key $\mathcal{SK}_{\mathsf{id}_k} = (K_1, \ldots, K_5, D, \mathsf{ktag})$.

If $Z_2 = (x_1 + x_2)H_2$, we have

$$\begin{aligned} K_1 &= \alpha P_2 + r'V_2 - a(x_1 + x_2)H_2 \\ &= \alpha P_2 - ar'H_2 - a(x_1 + x_2)H_2 \\ &= \alpha P_2 - (r' + x_1 + x_2)aH_2 \\ &= \alpha P_2 + rV_2 \end{aligned}$$

and

$$\begin{aligned} K_2 &= K'_2 + Z_2 + y'_v(x_1 P_2) \\ &= r'V'_2 - z'P_2 + (x_1 + x_2)H_2 + y'_v(x_1 P_2) \\ &= r'H_2 + r'y'_v P_2 - z'P_2 + (x_1 + x_2)H_2 + y'_v(x_1 P_2) + y'_v x_2 P_2 - y'_v x_2 P_2 \\ &= (r' + x_1 + x_2)H_2 + (r' + x_1 + x_2)y'_v P_2 - (z' + y'_v x_2)P_2 \\ &= rV'_2 - zP_2 \end{aligned}$$

indicating that $\mathcal{SK}_{\mathsf{id}_k}$ is a normally distributed key and so $\mathcal{B}$ perfectly simulates Game$_{k-1}$. When Z_2 is a random element of $\mathbb{G}_2$, it could be expressed as $Z_2 = (x_1 + x_2)H_2 + cP_2$. Then it is easy to see that $\mathcal{SK}_{\mathsf{id}_k}$ would be a semi-functional key for id_k with $\gamma = c$ in which case $\mathcal{B}$ is simulating Game$_k$.

Challenge: $\mathcal{B}$ receives the challenge identity id^* and two messages M_0 and M_1 from $\mathcal{A}$. It chooses $\beta \in \{0, 1\}$ at random. It cannot generate a semi-functional ciphertext for the challenge identity without knowledge of bV'_1 but setting $\mathsf{ctag}^* = \lambda\mathsf{id}^* + \nu$ enables it to do so. Since λ and ν are hidden from the attacker and $\lambda x + \nu$ is a pairwise independent function for the choice of λ and ν, both ctag^* and ktag will be independently and uniformly distributed as required. $\mathcal{B}$ first generates a normal ciphertext $\mathcal{C}' = (C'_0, C'_1, C'_2, C'_3, C'_4, C'_5, E'_1, E'_2, \mathsf{ctag}^*)$ with randomisers s, t' and $\mathsf{ctag}^* = \lambda \mathsf{id}^* + \nu$ using the **Encrypt** algorithm. Then it picks $\mu \in \mathbb{Z}_p$ at random and modifies $\mathcal{C}'$ as follows.

$$C_0 = C'_0,\ C_1 = C'_1,\ C_2 = C'_2 + \mu F_1,\ C_3 = C'_3 + \mu P_1,\ C_4 = C'_4 - \mu H_1 - \mu y'_v P_1$$

$$C_5 = C'_5 + \mu y_w H_1 - \mu y'_v F_1,\ E_1 = E'_1 + \mu(y_q \mathsf{id}^* + y_w \mathsf{ctag}^* + y_u) H_1,\ E_2 = E'_2 + \mu H_1$$

Since the simulator does not know bH_1 it has to construct C_5 by setting $tP_1 = t'P_1 + \mu H_1$ implicitly. We need only verify that C_5 is well-formed; rest of the elements are constructed according to the original semi-functional algorithms.

$$\begin{aligned}
C_5 &= C'_5 + \mu y_w H_1 - \mu y'_v F_1 \\
&= -b\tau s P_1 + t' W_1 + \mu y_w H_1 - \mu y'_v F_1 \\
&= -b\tau s P_1 + t'(F_1 + y_w P_1) + \mu y_w H_1 - \mu y'_v F_1 \\
&= -b\tau s P_1 + t' b P_1 + t' y_w P_1 + \mu y_w H_1 - \mu y'_v F_1 + b\mu H_1 - b\mu H_1 \\
&= -b\tau s P_1 + b(t'P_1 + \mu H_1) + y_w(t'P_1 + \mu H_1) - b\mu(y'_v P_1 + H_1) \\
&= -b\tau s P_1 + t W_1 - b\mu V'_1
\end{aligned}$$

$\mathcal{B}$ returns $\mathcal{C}^* = (C_0, C_1, C_2, C_3, C_4, C_5, E_1, E_2, \mathsf{ctag}^*)$ to $\mathcal{A}$.

In the key extraction phase, in order to generate a semi-functional ciphertext for id_k, the simulator must be able to compute bV'_1 or create $t(\mathsf{id}_k Q_1 + \mathsf{ctag} W_1 + U_1) = (\mathsf{ctag} - \lambda \mathsf{id}_k - \nu)(t'F_1 + b\mu H_1) + (y_q \mathsf{id}_k + y_w \mathsf{ctag} + y_u)(t'P_1 + \mu H_1)$ which is possible only when $\mathsf{ctag} = \lambda \mathsf{id}_k + \nu$. Decryption of this ciphertext with $\mathcal{SK}_{\mathsf{id}_k}$ will fail unconditionally and hence the simulator gains no information. Now if $\mathcal{A}$ is able to distinguish between Game_{k-1} and Game_k then it is able to decide whether $\mathcal{SK}_{\mathsf{id}_k}$ is normal or semi-functional. In this case $\mathcal{B}$ can solve the DLin problem with advantage

$$\mathsf{Adv}^{\mathcal{B}}_{\text{DLin}} = |\Pr[X_{k-1}] - \Pr[X_k]| = \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_k} = \varepsilon$$

□

Lemma 3. *If there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\text{Game}_q} - \mathsf{Adv}^{\mathcal{A}}_{final} = \varepsilon$, then we can build an algorithm $\mathcal{B}$ having advantage ε in breaking the* DBDH *assumption.*

Proof. $\mathcal{B}$ receives $(P_1, xP_1, aP_1, sP_1, P_2, xP_2, aP_2, sP_2, Z)$ as an instance of the DBDH problem.

Setup: With $b, y_v, y'_v, y_q, y_w, y_u$ chosen at random from $\mathbb{Z}_p$, $\mathcal{B}$ sets the parameters as

$$P_1 = P_1, P_2 = P_2, abP_1 = b(aP_1), V_2 = y_v P_2, V'_2 = y'_v P_2,\ \tau P_1 = y_v P_1 + y'_v(aP_1)$$

$$Q_1 = y_q P_1, W_1 = y_w P_1, U_1 = y_u P_1, e(P_1, P_2)^{b\alpha} = e(xP_1, aP_2)^b$$

implicitly setting $a = a$, $\alpha = xa$ and $\tau = y_v + a y'_v$. The remaining parameters can be computed easily. $\mathcal{B}$ returns $\mathcal{PP}$ to $\mathcal{A}$.

Phases 1 and 2: When $\mathcal{A}$ asks for the secret key for the i'th identity id_i, $\mathcal{B}$ chooses at random $r_1, r_2, z', \mathsf{ktag}, \gamma' \in \mathbb{Z}_p$ with $r = r_1 + r_2$. It implicitly sets $\gamma' = x - \gamma$ and $z = z' + x$. It then computes a semi-functional key for id_i as follows.

$$K_1 = \gamma'(aP_2) + rV_2 = xaP_2 - a\gamma P_2 + rV_2 = \alpha P_2 + rV_2 - a\gamma P_2$$

$$K_2 = rV_2' - z'P_2 - \gamma' P_2 = rV_2' - z'P_2 - xP_2 + \gamma P_2 = rV_2' - zP_2 + \gamma P_2$$

$$K_3 = bz'P_2 + b(xP_2) = bzP_2$$

$$K_4 = br_2P_2,\ K_5 = r_1P_2$$

$$D = r_1(\mathsf{id}_i Q_2 + \mathsf{ktag} W_2 + U_2).$$

Observe that $\mathcal{B}$ cannot create a normal key without knowing α.

Challenge: $\mathcal{B}$ receives the challenge identity id^* and two messages M_0 and M_1 from $\mathcal{A}$. It chooses $\beta \in \{0,1\}$ and $\mathsf{ctag}^*, \mu', t \in \mathbb{Z}_p$ at random and generates a semi-functional challenge ciphertext as follows. Here $\mathcal{B}$ implicitly sets $\mu' = \mu + as$.

$$C_0 = M_\beta \cdot Z^b,\ C_1 = b(sP_1)$$

$$C_2 = b\mu' P_1 = basP_1 + b\mu P_1, C_3 = \mu' P_1 = asP_1 + \mu P_1$$

$$C_4 = -y_v(sP_1) - \mu' y_v' P_1 = -y_v sP_1 - asy_v' P_1 - \mu y_v' P_1 = -\tau s P_1 - \mu V_1'$$

$$C_5 = -by_v(sP_1) - b\mu' y_v' P_1 + tW_1 = -by_v sP_1 - basy_v' P_1 - b\mu y_v' P_1 + tW_1 = -\tau s P_1 + tW_1 - \mu V_1'$$

$$E_1 = t(\mathsf{id}^* Q_1 + \mathsf{ctag}^* W_1 + U_1),\ E_2 = tP_1$$

The challenge ciphertext $\mathcal{C}^* = (C_0, C_1, C_2, C_3, C_4, C_5, E_1, E_2, \mathsf{ctag}^*)$ is returned to $\mathcal{A}$. If Z equals $e(P_1,P_2)^{xas}$ then $\mathcal{C}^*$ will be a semi-functional encryption of M_β; if Z is a random element of $\mathbb{G}_T$ then $\mathcal{C}^*$ will be a semi-functional encryption of a random message. If $\mathcal{A}$ can identify whether the game simulated was Game_q or Game_{final}, then $\mathcal{B}$ will be able to decide whether $Z = e(P_1,P_2)^{xas}$ or not and hence break the DBDH assumption with advantage

$$\mathsf{Adv}^{\mathcal{B}}_{\text{DBDH}} = |\Pr[X_q] - \Pr[X_{final}]| = \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_q} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{final}} = \varepsilon$$

□

Let $\varepsilon_{\text{DDH1}} = \max_{\mathcal{B}} \mathsf{Adv}^{\mathcal{B}}_{\text{DDH1}}$ where the maximum is taken over all PPT algorithms $\mathcal{B}$. Similarly define $\varepsilon_{\text{DLin}}$ and $\varepsilon_{\text{DBDH}}$.

Theorem 2. *If the* DDH1, DLin *and* DBDH *assumptions hold, then no PPT adversary making at most q key extract queries can break the security of Scheme 1.*

Proof. Using lemmas 1, 2 and 3, we have for any polynomial time attacker $\mathcal{A}$,

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}}_{\text{Scheme}-1} &\le |\Pr[X_{real}] - \Pr[X_0]| + \sum_{k=1}^{q} (|\Pr[X_{k-1}] - \Pr[X_k]|) + |\Pr[X_q] - \Pr[X_{final}]| \\ &= \varepsilon_{\text{DDH1}} + q\varepsilon_{\text{DLin}} + \varepsilon_{\text{DBDH}}\end{aligned}$$

which is negligible in the security parameter κ. □

E Security Proof for Scheme 6

Define Game$_{real}$, Game$_k$ (for $0 \le k \le q$) and Game$_{final}$ as before.

Lemma 4. *If there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_{real}} - \mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_0} = \varepsilon$, then we can build an algorithm $\mathcal{B}$ having advantage ε in solving the* DDH1 *problem.*

Proof. The algorithm $\mathcal{B}$ receives $(P_1, sP_1, aP_1, P_2, Z_1)$ as an instance of DDH1. We describe how it will simulate each phase in the security game.

Setup: $\mathcal{B}$ chooses random elements $\alpha, y_v, y'_v, y_q, y_w, y_u$ from $\mathbb{Z}_p$ and sets the parameters as follows.

$$P_1 = P_1, sP_1 = sP_1, aP_1 = aP_1, Q_1 = y_q P_1, W_1 = y_w P_1, U_1 = y_u P_1$$

$$P_2 = P_2, V_2 = y_v P_2, V'_2 = y'_v P_2$$

This implicitly sets $\tau = y_v + a y'_v$. Using this, the element τP_1 can be computed as $y_v P_1 + y'_v(aP_1)$. The simulator computes the remaining parameters using α and gives the following public parameters to $\mathcal{A}$.

$$\mathcal{PP} = \{\mathcal{G}, P_1, P_2, aP_1, \tau P_1, Q_1, W_1, U_1, e(P_1, P_2)^{b\alpha}\}$$

Phase 1: $\mathcal{A}$ makes a number of key extract queries. $\mathcal{B}$ knows the master secret and using that it returns a normal key for every key extract query made by $\mathcal{A}$.

Challenge: $\mathcal{B}$ receives the target identity id^* and two messages M_0 and M_1 from $\mathcal{A}$. It chooses $\beta \in \{0, 1\}$ at random. To encrypt M_β, $\mathcal{B}$ chooses ctag^* at random from $\mathbb{Z}_p$ and computes the ciphertext elements as follows.

$$C_0 = M_\beta \cdot e(sP_1, P_2)^\alpha$$

$$C_1 = sP_1,\ C_2 = Z_1,\ C_3 = -y_v(sP_1) - y'_v Z_1 + y_w(sP_1)$$

$$E = (\mathsf{id}^* y_q + \mathsf{ctag}^* y_w + y_u)(sP_1)$$

$\mathcal{B}$ returns $\mathcal{C}^* = (C_0, C_1, C_2, C_3, E, \mathsf{ctag}^*)$ to $\mathcal{A}$.

If $Z_1 = asP_1$ then the challenge ciphertext is normal; otherwise if Z_1 is a random element of $\mathbb{G}_1$ i.e., $Z_1 = (as + c)P_1$ then the ciphertext is semi-functional with $\mu = c$.

Note that, to check whether $\mathcal{C}^*$ is semi-functional or not, $\mathcal{B}$ itself could try to decrypt it with a semi-functional key for id^*. However since aP_2 is not known to $\mathcal{B}$, it cannot create such a key.

Phase 2: As in first phase, $\mathcal{B}$ returns a normal key for every query.

Guess: The adversary returns its guess β' to $\mathcal{B}$.

If $\mathcal{C}^*$ is normal then $\mathcal{B}$ simulates Game$_{real}$ and if it is semi-functional $\mathcal{B}$ simulates Game$_0$. Therefore if $\mathcal{A}$ is able to distinguish between Game$_{real}$ and Game$_0$ i.e., if $\beta = \beta'$, then the simulator can solve the DDH1 problem with advantage

$$\mathsf{Adv}^{\mathcal{B}}_{\mathrm{DDH1}} = |\Pr[X_{real}] - \Pr[X_0]| = \mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_{real}} - \mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_0} = \varepsilon$$

□

Lemma 5. *If there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_k} = \varepsilon$, then we can build an algorithm $\mathcal{B}$ having advantage ε in breaking the assumption* DDH2v.

Proof. Let $(P_1, dP_1, dzP_1, zx_1P_1, P_2, dP_2, x_1P_1, x_2P_2, Z_2)$ be the instance of DDH2v that $\mathcal{B}$ receives.

Setup: $\mathcal{B}$ chooses random elements $a, \alpha, \lambda, \nu, y'_v, y_q, y_u, y_w \in_R \mathbb{Z}_p$ and sets the parameters as follows. $P_1 = P_1, P_2 = P_2$, $Q_2 = -\lambda(dP_2) + y_qP_2$, $U_2 = -\nu(dP_2) + y_uP_2$, $W_2 = dP_2 + y_wP_2$, $V_2 = -a(x_1P_2)$ and $V'_2 = x_1P_2 + y'_vP_2$ setting $\tau = ay'_v$ using which one can compute $\tau P_1 = ay'_vP_1$. The public parameters Q_1, W_1, U_1 can be computed since $\mathcal{B}$ has dP_1. The remaining parameters required to provide $\mathcal{PP}$ to $\mathcal{A}$ are computed using a, α and other elements of the problem instance.

Phases 1 and 2: The key extraction queries for identities $\mathsf{id}_1, \ldots, \mathsf{id}_q$ are answered in the following way. For $i < k$, a semi-functional key is returned and for $i > k$ a normal key is returned. For $i = k$, a normal key K'_1, K'_2, K'_3, D' is generated using randomiser $r' \in_R \mathbb{Z}_p$, $\mathsf{ktag} = \lambda\mathsf{id}_k + \nu$ and then modified as:

$$K_1 = K'_1 - aZ_2,\ K_2 = K'_2 + Z_2 + y'_v(x_2P_2),\ K_3 = K'_3 + x_2P_2$$
$$D = D + (y_q\mathsf{id} + y_w\mathsf{ktag} + y_u)(x_2P_2)$$

thus implicitly setting $r = r' + x_2$. Since dx_2P_2 is not known to $\mathcal{B}$ it can create D only when $\mathsf{ktag} = \lambda\mathsf{id}_k + \nu$. If $Z_2 = x_1x_2P_2$ then the key for id_k will be normal and otherwise it will be semi-functional with $\gamma = c$ where $Z_2 = (x_1x_2 + c)P_2$. Note that a semi-functional ciphertext for id_k with any value of ctag cannot be generated without the knowledge of dx_1zP_1 which is neither available from the assumption nor can be computed by $\mathcal{B}$. This rules out the obvious way of checking whether the key for id_k is semi-functional or not.

Challenge: $\mathcal{B}$ receives two messages M_0, M_1 and a challenge identity id^* during the challenge phase. It chooses $\beta \in_R \{0,1\}$, generates a normal ciphertext with randomiser $s' \in_R \mathbb{Z}_p$, $\mathsf{ctag}^* = \lambda\mathsf{id}^* + \nu$ and changes the ciphertext elements as follows. Since λ and ν are chosen independently and uniformly at random, the function $\lambda X + \nu$ is a pairwise independent function. This causes the tag values of the challenge ciphertext and the k-th key to appear properly distributed from the adversary's view.

$$C_0 = C'_0 \cdot e(zx_1P_1, P_2)$$
$$C_1 = C'_1 + x_1zP_1,\ C_2 = C'_2 + a(zx_1P_1) + dzP_1,\ C_3 = C'_3 - ay'_v(zx_1P_1) + y_w(zx_1P_1) - y'_v(dzP_1)$$
$$E = E' + (y_q\mathsf{id} + \mathsf{ctag}^*y_w + y_u)(zx_1P_1)$$

setting $s = s' + zx_1$ and $\mu = dz$. It is easy to check that C_3 is well-formed.

$$\begin{aligned} C_3 &= C'_3 - ay'_v(zx_1P_1) + y_w(zx_1P_1) - y'_v(dzP_1) \\ &= -ay'_v(s' + zx_1)P_1 + s'W_1 + y_w(zx_1P_1) + dzx_1P_1 - dzx_1P_1 - y'_vdzP_1 \\ &= -ay'_vsP_1 + s'W_1 + zx_1(dP_1 + y_wP_1) - dz(x_1P_1 + y'_vP_1) \\ &= -ay'_vsP_1 + sW_1 - \mu V'_1 \end{aligned}$$

Now $\mathcal{A}$ will be able to distinguish between Game_{k-1} and Game_k if it can decide whether $\mathcal{SK}_{\mathsf{id}_k}$ is normal or semi-functional. In this case $\mathcal{B}$ can break the assumption DDH2v with advantage

$$\mathsf{Adv}^{\mathcal{B}}_{\mathrm{DDH2v}} = |\Pr[X_{k-1}] - \Pr[X_k]| = \mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_k} = \varepsilon$$

□

Lemma 6. *If there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_q} - \mathsf{Adv}^{\mathcal{A}}_{final} = \varepsilon$, then we can build an algorithm $\mathcal{B}$ having advantage ε in breaking the* DBDH *assumption.*

Proof. $\mathcal{B}$ receives $(P_1, xP_1, aP_1, sP_1, P_2, xP_2, aP_2, sP_2, Z)$ as an instance of the DBDH problem.

Setup: With $b, y_v, y'_v, y_q, y_w, y_u$ chosen at random from $\mathbb{Z}_p$, $\mathcal{B}$ sets the parameters as

$$P_1 = P_1, P_2 = P_2, aP_1 = aP_1, V_2 = y_v P_2, V'_2 = y'_v P_2,\ \tau P_1 = y_v P_1 + y'_v(aP_1)$$
$$Q_1 = y_q P_1, W_1 = y_w P_1, U_1 = y_u P_1, e(P_1, P_2)^{b\alpha} = e(xP_1, aP_2)$$

implicitly setting $a = a$, $\alpha = xa$ and $\tau = y_v + ay'_v$. The remaining parameters can be computed easily. $\mathcal{B}$ returns $\mathcal{PP}$ to $\mathcal{A}$.

Phases 1 and 2: When $\mathcal{A}$ asks for the secret key for the i'th identity id_i, $\mathcal{B}$ chooses at random $r, \mathsf{ktag}, \gamma' \in \mathbb{Z}_p$ implicitly setting $\gamma' = x - \gamma$. It then computes a semi-functional key for id_i as follows.

$$K_1 = \gamma'(aP_2) + rV_2 = xaP_2 - a\gamma P_2 + rV_2 = \alpha P_2 + rV_2 - a\gamma P_2$$
$$K_2 = rV'_2 - \gamma' P_2 = rV'_2 + xP_2 + \gamma P_2 = rV'_2 + \gamma P_2$$
$$K_3 = rP_2, D = r(\mathsf{id}_i Q_2 + \mathsf{ktag} W_2 + U_2).$$

Observe that $\mathcal{B}$ does not know α and hence cannot create a normal key.

Challenge: $\mathcal{B}$ receives the challenge identity id^* and two messages M_0 and M_1 from $\mathcal{A}$. It chooses $\beta \in \{0,1\}$ and $\mathsf{ctag}^*, \mu', t \in \mathbb{Z}_p$ at random and generates a semi-functional challenge ciphertext as follows. Here $\mathcal{B}$ implicitly sets $\mu' = \mu + as$.

$$C_0 = M_\beta \cdot Z^b,$$
$$C_1 = sP_1, C_2 = \mu' P_1 = asP_1 + \mu P_1$$
$$C_3 = -y_v(sP_1) - \mu' y'_v P_1 + y_w(sP_1) = -y_v sP_1 - asy'_v P_1 - \mu y'_v P_1 + sW_1 = -\tau sP_1 - \mu V'_1 + sW_1$$
$$E = (y_q \mathsf{id}^* + y_w \mathsf{ctag}^* + y_u)(sP_1)$$

The challenge ciphertext $\mathcal{C}^* = (C_0, C_1, C_2, C_3, E, \mathsf{ctag}^*)$ is returned to $\mathcal{A}$. If $Z = e(P_1, P_2)^{xas}$ then $\mathcal{C}^*$ will be a semi-functional encryption of M_β; if Z is a random element of $\mathbb{G}_T$ then $\mathcal{C}^*$ will be a semi-functional encryption of a random message. If $\mathcal{A}$ can identify whether the game simulated was Game_q or Game_{final}, then $\mathcal{B}$ will be able to decide whether $Z = e(P_1, P_2)^{xas}$ or not and hence break the DBDH assumption with advantage

$$\mathsf{Adv}^{\mathcal{B}}_{\mathrm{DBDH}} = |\Pr[X_q] - \Pr[X_{final}]| = \mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_q} - \mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_{final}} = \varepsilon$$

□

Theorem 3. *If the* DDH1, DDH2v *and* DBDH *assumptions hold, then no polynomial time adversary $\mathcal{A}$ making at most q key extraction queries can break the security of Scheme 6.*

Proof. Using lemmas 4, 5 and 6, we have for any polynomial time attacker $\mathcal{A}$,

$$\mathsf{Adv}^{\mathcal{A}}_{\mathrm{Scheme-6}} \leq |\Pr[X_{real}] - \Pr[X_0]| + \sum_{k=1}^{q} (|\Pr[X_{k-1}] - \Pr[X_k]|) + |\Pr[X_q] - \Pr[X_{final}]|$$
$$= \varepsilon_{\mathrm{DDH1}} + q\varepsilon_{\mathrm{DDH2v}} + \varepsilon_{\mathrm{DBDH}}$$

which is negligible in the security parameter κ. □

F Schemes 2 to 5

We follow a compact notation to denote normal and semi-functional ciphertexts and keys. In the description of ciphertexts and keys, certain group elements are shown in curly brackets $\{\}$. These constitute the semi-functional components.

F.1 Scheme 2

$P_1, Q_1, W_1, U_1 \in \mathbb{G}_1$ and $P_2 \in G_2$ are random generators. Let $Q_2, W_2, U_2 \in \mathbb{G}_2$ with $Q_2 \sim Q_1$, $W_2 \sim W_1$ and $U_2 \sim U_1$. Choose a_1, a_2, v, v', v'' be chosen at random from $\mathbb{Z}_p$ and define $V_2 = vP_2, V_2' = v'P_2, V_2'' = v''P_2$ be chosen at random. Define $\tau_1 = v + a_1 v'$ and $\tau_2 = v + a_2 v''$ so that $\tau_1 P_2 = V_2 + a_1 V_2'$ and $\tau_2 P_2 = V_2 + a_2 V_2''$.

The public parameters and master secret are given by

$$\mathcal{PP} = \{\mathcal{G}, P_1, a_1P_1, a_2P_1, \tau_1P_1, \tau_2P_1, Q_1, W_1, U_1, e(P_1, P_2)^{\alpha}\}$$

$$\mathcal{MSK} = \{P_2, \alpha P_2, V_2, V_2', V_2'', Q_2, W_2, U_2\}$$

The randomisers for the ciphertext are $s_1, s_2, t \in \mathbb{Z}_p$ with $s = s_1 + s_2$. $V_1'' \in \mathbb{G}_1$ has the same discrete log to base P_1 as V_2'' to base P_2.

Ciphertext:
$C_0 = M \cdot e(P_1, P_2)^{\alpha s}$
$C_1 = sP_1,\ C_2 = a_1 s_1 P_1,\ C_3 = a_2 s_2 P_1\ \{+a_2 \mu P_1\},\ C_4 = -(\tau_1 s_1 + \tau_2 s_2)P_1 + tW_1\ \{-a_2 \mu V_1''\}$
$E_1 = t(\mathsf{id}Q_1 + \mathsf{ctag}W_1 + U_1),\ E_2 = tP_1$

Key:
$K_1 = \alpha P_2 + rV_2\ \{-a_1 a_2 \gamma P_2\},\ K_2 = rV_2'\ \{+a_2 \gamma P_2\},\ K_3 = rV_2''\ \{+a_1 \gamma P_2\},\ K_4 = rP_2$
$D = r(\mathsf{id}Q_2 + \mathsf{ktag}W_2 + U_2)$

Security: With the DLin assumption, it is possible to show that Game_{real} and Game_0 are indistinguishable. We give a brief sketch of the proof. Let $(P_1, F_1, H_1, x_1P_1, x_2F_1, P_2, F_2, H_2, Z_1)$ be an instance of DLin that the simulator $\mathcal{B}$ receives. It chooses $\alpha, y_q, y_w, y_u, y_v, y_v', y_v'' \in_R \mathbb{Z}_p$ and constructs the parameters as $P_1 = P_1, a_1P_1 = F_1, a_2P_1 = H_1, Q_1 = y_qP_1, W_1 = y_wP_1, U_1 = y_uP_1, V_2 = y_vP_2, V_2' = y_v'P_2, V_2'' = y_v''P_2$, implicitly setting $\tau_1 = y_v + a_1 y_v'$ and $\tau_2 = y_v + a_1 y_v''$. Since $\mathcal{B}$ has the master secret it can create a normal key for any identity. It answers all the key extraction queries with normal keys. In the challenge phase it receives two messages M_0, M_1 and an identity id^*. It then chooses $\beta \in_R \{0, 1\}$. A normal ciphertext $C_0', \ldots, C_4', E_1', E_2', \mathsf{ctag}$ for identity id^* and message M_β is first generated using randomisers s_1', s_2', t and then modified as follows:

$$C_0 = C_0' \cdot e(x_1P_1, P_2)^{\alpha}, C_1 = C_1' + x_1P_1, C_2 = C_2' - x_2F_1$$

$$C_3 = C_3' + Z_1, C_4 = C_4' - y_vx_1P_1 + y_v'x_2F_1 - y_v''Z_1, E_1 = E_1', E_2 = E_2'$$

implicitly setting $s_1 = s_1' - x_2$, $s_2 = s_2' + x_1 + x_2$ and $s = s_1 + s_2 = s' + x_1$. If $Z_1 = (x_1 + x_2)H_1$ then the ciphertext is normal; otherwise $Z_1 = (x_1 + x_2 + c)H_1$ for some $c \in_R \mathbb{Z}_p$ whence the ciphertext is semi-functional with μ being set to c.

Now let's take a look at the second reduction and it's proof using DDH2 and DLin assumptions.

First consider the DDH2 assumption. Let $(P_1, P_2, x_1P_2, x_2P_2, Z_2)$ be an instance of DDH2. As mentioned earlier, except for P_1 such an instance consists of elements of $\mathbb{G}_2$. The $\mathcal{PP}$, on the other hand, consists entirely of elements of $\mathbb{G}_1$. There is no way that the $\mathcal{PP}$ can be based on the given instance and so getting a reduction is not possible. Now consider the DLin assumption. In this scheme, the randomiser r is not split. As such there is no way to base this reduction on the DLin assumption.

F.2 Scheme 3

Random generators $P_1, Q_1, W_1, U_1 \in \mathbb{G}_1$ and $P_2 \in G_2$ are chosen. Let $\alpha, a, b \in_R \mathbb{Z}_p$ and $Q_2, W_2, U_2 \in \mathbb{G}_2$ with $Q_2 \sim Q_1$, $W_2 \sim W_1$ and $U_2 \sim U_1$. Also, let $V_2, V_2' \in_R \mathbb{G}_2$ and $\tau \in \mathbb{Z}_p$ such that $\tau P_2 = V_2 + aV_2'$.

The public parameters and master secret are given by

$$\mathcal{PP} = \{\mathcal{G}, P_1, P_2, bP_1, abP_1, \tau P_1, b\tau P_1, Q_1, W_1, U_1, e(P_1, P_2)^{b\alpha}\}$$

$$\mathcal{MSK} = \{\alpha P_2, bP_2, V_2, V_2', Q_2, W_2, U_2\}$$

V_1' will have the usual meaning. Randomisers for the ciphertext and the key are s, t and r_1, r_2 respectively with $r = r_1 + r_2$.

Ciphertext
$C_0 = M \cdot e(P_1, P_2)^{b\alpha s}$
$C_1 = bsP_1$, $C_2 = basP_1\{+b\mu P_1\}$, $C_3 = -\tau sP_1 \{-\mu V_1'\}$, $C_4 = -b\tau sP_1 + tW_1 \{-b\mu V_1'\}$
$E_1 = t(\mathsf{id}Q_1 + \mathsf{ctag}W_1 + U_1)$, $E_2 = tP_1$

Key
$K_1 = \alpha P_2 + rV_2 \{-a\gamma P_2\}$, $K_2 = rV_2' \{+\gamma P_2\}$, $K_3 = br_2P_2$, $K_4 = r_1P_2$
$D = r_1(\mathsf{id}Q_2 + \mathsf{ktag}W_2 + U_2)$

Security: The first reduction goes through with the DDH1 assumption. We provide a brief sketch. Suppose that $(P_1, aP_1, sP_1, P_2, Z_1)$ is the instance of DDH1. It simulates the game in the following way – choose $\alpha, b, y_v, y_v', y_q, y_u, y_w \in_R \mathbb{Z}_p$ and set $P_1 = P_1, aP_1 = aP_1, Q_1 = y_qP_1, W_1 = y_wP_1, U_1 = y_uP_1, P_2 = P_2V_2 = y_vP_2, V_2' = y_v'P_2$. Construct the challenge ciphertext as

$$C_0 = M_\beta \cdot e(sP_1, P_2)^{b\alpha}, C_1 = b(sP_1), C_2 = bZ_1, C_3 = -y_v(sP_1) - y_v'Z_1, C_4 = -by_v(sP_1) - by_v'Z_1 + tW_1.$$

Computing E_1 and E_2 is straightforward. Note that the randomiser s comes from the assumption.

For the second reduction, we cannot rely on DDH2 assumption for the same reason as that for Scheme 2. In this case, there is a split of the randomiser r. So, one may hope to base the reduction on the DLin problem.

Consider an instance of the DLin assumption: $(P_1, F_1, H_1, P_2, F_2, H_2, x_1P_2, x_2F_2, Z_2)$. The simulator $\mathcal{B}$ chooses $\alpha, b, y_v', y_q, y_u, y_w \in_R \mathbb{Z}_p$ and computes the parameters as $P_1 = P_1, P_2 = P_2, bP_1 = F_1, V_2 = -aH_2, V_2' = aH_2 + y_v'P_2$ so that Z_2 could be embedded in K_1 and K_2. Observe that

$\tau = ay'_v$. We can successfully construct K_1, K_3 and K_4 setting $r_1 = r'_1 + x_1$, $r_2 = r'_2 + x_2$ and $r = r' + x_1 + x_2$, but composing K_2 with the same r is impossible. Any other way of using DLin assumption also results in failure.

F.3 Scheme 4

The public parameters and master secret are given by

$$\mathcal{PP} = \{\mathcal{G}, P_1, P_2, aP_1, abP_1, \tau P_1, Q_1, W_1, U_1, e(P_1, P_2)^{\alpha}\}$$

$$\mathcal{MSK} = \{\alpha P_2, bP_2, V_2, V'_2, Q_2, W_2, U_2\}$$

Randomisers for the ciphertext and the key are s, t and r, z respectively.

Ciphertext
$C_0 = M \cdot e(P_1, P_2)^{\alpha s}, C_1 = sP_1, C_2 = asP_1\{+\mu P_1\}, C_3 = basP_1\{+b\mu P_1\}, C_4 = -\tau sP_1 + tW_1\{-\mu V'_1\}$
$E_1 = t(\mathsf{id}Q_1 + \mathsf{ctag}W_1 + U_1),\ E_2 = tP_1$

Key
$K_1 = \alpha P_2 + rV_2\ \{-a\gamma P_2\},\ K_2 = rV'_2 - bzP_2\ \{+\gamma P_2\},\ K_3 = zP_2,\ K_4 = rP_2$
$D = r(\mathsf{id}Q_2 + \mathsf{ktag}W_2 + U_2)$

Here, the first reduction goes through with the DDH1 assumption. The argument is similar to that of Scheme 3. As before, the second reduction cannot be based on the DDH2 assumption. Since there is no split of the randomiser r, there is no way to base the second reduction on the DLin assumption.

F.4 Scheme 5

Let $V_2, V'_2 \in \mathbb{G}_2$ and $\tau P_2 = V_2 + aV'_2$. V'_1 will have the usual meaning. The $\mathcal{PP}$ and $\mathcal{MSK}$ are given by

$$\mathcal{PP} = \{\mathcal{G}, P_1, P_2, aP_1, \tau P_1, Q_1, W_1, U_1, e(P_1, P_2)^{\alpha}\}$$

$$\mathcal{MSK} = \{\alpha P_2, V_2, V'_2, Q_2, W_2, U_2\}$$

For the ciphertext and the key, $s, t \in \mathbb{Z}_p$ and $r \in \mathbb{Z}_p$ respectively are chosen as the randomisers.

Ciphertext
$C_0 = M \cdot e(P_1, P_2)^{\alpha s},\ C_1 = sP_1,\ C_2 = asP_1\ \{+\mu P_1\},\ C_3 = -\tau sP_1 + tW_1\ \{-\mu V'_1\}$
$E_1 = t(\mathsf{id}Q_1 + \mathsf{ctag}W_1 + U_1),\ E_2 = tP_1$

Key
$K_1 = \alpha P_2 + rV_2\ \{-a\gamma P_2\},\ K_2 = rV'_2\ \{+\gamma P_2\},\ K_3 = rP_2$
$D = r(\mathsf{id}Q_2 + \mathsf{ktag}W_2 + U_2)$

The first security reduction can be based on DDH1 assumption but the second reduction does not hold with either DDH2 or the DLin assumption.

G Extending Scheme 6 to a HIBE Scheme

We provide a brief description of the HIBE scheme. The nature of the extension is completely analogous to the way the IBE in [21] has been extended to a HIBE.

The setting is of an asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ as before. Messages are elements of $\mathbb{G}_T$. Identities are tuples of $\mathbb{Z}_p$ of lengths varying from 1 to some maximum size n. As in [21], we assume that if two identities agree on some component, then they agree on all previous components. The HIBE scheme that we describe below is an extension of Scheme 6. We describe the various functionalities of this scheme.

G.1 Construction

Let a, v, v' be random elements of $\mathbb{Z}_p$. Set $V_2 = vP_2$, $V_2' = v'P_2$ and $\tau = v + av'$ so that $\tau P_2 = V_2 + aV_2'$. Let W_1 be a random element of $\mathbb{G}_1$ and let $W_2 \in \mathbb{G}_2$ be such that $W_1 \sim W_2$. For $1 \le i \le n$, let $Q_{1,i}, U_{1,i}$ be random elements of $\mathbb{G}_1$ and $Q_{2,i}, U_{2,i}$ be elements of $\mathbb{G}_2$ such that $Q_{1,i} \sim Q_{2,i}$ and $U_{1,i} \sim U_{2,i}$.

$\mathcal{PP}$: $(P_1, aP_1, \tau P_1, Q_{1,1}, \ldots, Q_{1,n}, W_1, U_{1,1}, \ldots, U_{1,n}, e(P_1, P_2)^{\alpha})$.
$\mathcal{MSK}$: $(P_2, \alpha P_2, V_2, V_2', Q_{2,1}, \ldots, Q_{2,n}, W_2, U_{2,1}, \ldots, U_{2,n})$.

Actually, only αP_2 is the master secret. The other components of $\mathcal{MSK}$ has to be provided with the decryption key so as to enable further key delegation.

Encrypt$(M, (\mathsf{id}_1, \ldots, \mathsf{id}_\ell), \mathcal{PP})$: Choose random s and $\mathsf{ctag}_1, \ldots, \mathsf{ctag}_\ell$ from $\mathbb{Z}_p$. $\mathcal{C}$ is

$$(C_0, C_1, C_2, C_3, E_1, \ldots, E_\ell, \mathsf{ctag}_1, \ldots, \mathsf{ctag}_\ell)$$

where the elements are defined as follows.

$C_0 = M \cdot e(P_1, P_2)^{\alpha s}$,
$C_1 = sP_1$, $C_2 = asP_1$, $C_3 = -\tau sP_1 + sW_1$,
$E_i = s(\mathsf{id}_i Q_{1,i} + \mathsf{ctag}_i W_1 + U_{1,i})$ for $1 \le i \le \ell$.

KeyGen$(\overrightarrow{\mathsf{id}} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell), \mathcal{MSK}, \mathcal{PP})$: Choose random $r_1, \ldots, r_\ell, \mathsf{ktag}_1, \ldots, \mathsf{ktag}_\ell$ from $\mathbb{Z}_p$ and set $r = r_1 + \cdots + r_\ell$. $\mathcal{SK}_{\overrightarrow{\mathsf{id}}}$ consists of the elements $(P_2, V_2, V_2', Q_{2,1}, \ldots, Q_{2,n}, W_2, U_{2,1}, \ldots, U_{2,n})$ along with the tuple $(K_1, K_2, K_{3,1}, \ldots, K_{3,\ell}, D_1, \ldots, D_\ell, \mathsf{ktag}_1, \ldots, \mathsf{ktag}_\ell)$ where the elements are defined as follows.

$K_1 = \alpha P_2 + rV_2$, $K_2 = rV_2'$
$K_{3,i} = r_i P_2$ for $i = 1, \ldots, \ell$,
$D_i = r_i(\mathsf{id}_i Q_{2,i} + \mathsf{ktag}_i W_2 + U_{2,i})$ for $i = 1, \ldots, \ell$.

The elements $(P_2, V_2, V_2', Q_{2,1}, \ldots, Q_{2,n}, W_2, U_{2,1}, \ldots, U_{2,n})$ are provided to enable further delegation.

Delegate($\overrightarrow{\mathsf{id}} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell), \mathcal{SK}_{\overrightarrow{\mathsf{id}}}, \mathsf{id}_{\ell+1}, \mathcal{PP}$): Other than the tags, all elements of $\mathcal{SK}_{\overrightarrow{\mathsf{id}}}$ corresponding to identity components at the previous levels are re-randomised. The components required for further delegation are not re-randomised.

Choose random $r_1, \ldots, r_d, r_{\ell+1}, \mathsf{ktag}_{\ell+1}$ from $\mathbb{Z}_p$ and set $r = r_1 + \cdots + r_\ell + r_{\ell+1}$. The new components of the key for the identity tuple of length $(\ell+1)$ are as follows.

$$K_{3,\ell+1} = r_{\ell+1}P_2,\ D_{\ell+1} = r_{\ell+1}(\mathsf{id}_i Q_{2,\ell+1} + \mathsf{ktag}_{\ell+1}W_2 + U_{2,\ell+1})P_2.$$

Re-randomisation of the previous components is done as follows.

$K_1 \leftarrow K_1 + rV_2,\ K_2 \leftarrow K_2 + rV_2'$
$K_{3,i} \leftarrow K_{3,i} + r_iP_2$ for $i = 1, \ldots, \ell$,
$D_i \leftarrow D_i + r_i(\mathsf{id}_i Q_{2,i} + \mathsf{ktag}_i W_2 + U_{2,i})$ for $i = 1, \ldots, \ell$.

Decrypt ($\mathcal{C}, \mathsf{id}, \mathcal{SK}_{\mathsf{id}}, \mathcal{PP}$): Decryption succeeds only when $\mathsf{ctag}_i \neq \mathsf{ktag}_i$ for $i = 1, \ldots, \ell$. First compute

$$A_1 = \prod_{i=1}^{\ell} \left(\frac{e(E_i, K_{3,i})}{e(C_1, D_i)} \right)^{1/(\mathsf{ctag}_i - \mathsf{ktag}_i)} = e(W_1, P_2)^{rs}.$$

Then compute

$$A_2 = e(C_1, K_1)e(C_2, K_2)e\left(C_3, \sum_{i=1}^{\ell} K_{3,i} \right) = e(P_1, P_2)^{\alpha s} e(W_1, P_2)^{rs}.$$

Unmask the message as $M = (C_0 \cdot A_1)/A_2$.

G.2 Security Proof

We first define semi-functional ciphertexts and keys for the HIBE.

Semi-functional ciphertext: Let $C_0', C_1', C_2', C_3', E_1', \ldots, E_d', \mathsf{ctag}_1, \ldots, \mathsf{ctag}_\ell$ be ciphertext elements normally generated by the **Encrypt** algorithm for message M and identity $\overrightarrow{\mathsf{id}} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$. Let V_1' be an element of $\mathbb{G}_1$ such that $V_1' \sim V_2'$. Choose $\mu \in \mathbb{Z}_p$ at random. The semi-functional ciphertext generation algorithm will modify the normal ciphertext as: $C_0 = C_0'$, $C_1 = C_1'$, $E_j = E_j'$ for $1 \leq j \leq \ell$ and

$$C_2 = C_2' + \mu P_1,\ C_3 = C_3' - \mu V_1'.$$

Semi-functional key: Let $K_1', K_2', K_{3,1}', \ldots, K_{3,\ell}', D_1', \ldots, D_\ell', \mathsf{ktag}_1, \ldots, \mathsf{ktag}_\ell$ be secret key components normally generated by the **KeyGen** algorithm for identity $\overrightarrow{\mathsf{id}} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$. Randomly choose $\gamma \in \mathbb{Z}_p$. The semi-functional key generation algorithm will modify the normal key as: $K_{3,j} = K_{3,j}'$, $D_j = D_j'$ for $1 \leq j \leq \ell$ and

$$K_1 = K_1' - a\gamma P_2,\ K_2 = K_2' + \gamma P_2.$$

For the security proof, we follow the path set out by [21]. Suppose that the attacker $\mathcal{A}$ makes at most q_R reveal queries and q_A create and delegate queries. Security is proved via a hybrid argument over a sequence of $q_R + 3$ games. Let Game_{real} be the actual HIBE security game defined in Section A.2. Game_0 is similar to Game_{real} except that the challenge ciphertext is semi-functional. In Game_k (for $1 \le k \le q_R$), the first k keys are semi-functional and the rest are normal. During the reduction, the keys are changed from normal to semi-functional just before they are revealed. Game_{final} is just like Game_{q_R} except that the challenge ciphertext is a semi-functional encryption of a random message. Define X_{real}, X_k for $0 \le k \le q_R$ and X_{final} as earlier. The following lemmas provide the indistinguishability arguments.

Lemma 7. *If there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{real}} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_0} = \varepsilon$, then we can build an algorithm $\mathcal{B}$ having advantage ε in solving the* DDH1 *problem.*

Proof. The algorithm $\mathcal{B}$ receives $(P_1, sP_1, aP_1, P_2, Z_1)$ as an instance of DDH1. We describe how it will simulate each phase in the security game.

Setup: $\mathcal{B}$ chooses random elements $\alpha, y_v, y'_v, y_{q_1}, \ldots, y_{q_n}, y_w, y_{u_1}, \ldots, y_{u_n}$ from $\mathbb{Z}_p$ and sets the parameters as follows.

$$P_1 = P_1, sP_1 = sP_1, aP_1 = aP_1, P_2 = P_2, V_2 = y_v P_2, V'_2 = y'_v P_2$$

$$W_i = y_w P_i, Q_{i,j} = y_{q_j} P_i, U_{i,j} = y_{u_j} P_i \text{ for } i = 1,2 \text{ and } 1 \le j \le n$$

This implicitly sets $\tau = y_v + a y'_v$. Using this, the element τP_1 can be computed as $y_v P_1 + y'_v(aP_1)$. The simulator computes the remaining parameters using α and gives the following public parameters to $\mathcal{A}$.

$$\mathcal{PP} = \{\mathcal{G}, P_1, P_2, aP_1, \tau P_1, Q_{1,1}, \ldots, Q_{1,n}, W_1, U_{1,1}, \ldots, U_{1,n}, e(P_1, P_2)^{b\alpha}\}$$

Phase 1: $\mathcal{A}$ makes a number of key extract queries. $\mathcal{B}$ knows the master secret and using that it returns a normal key for every key extract query made by $\mathcal{A}$.

Challenge: $\mathcal{B}$ receives the target identity $\overrightarrow{\mathsf{id}}^* = (\mathsf{id}^*_1, \ldots, \mathsf{id}^*_\ell)$ and two messages M_0 and M_1 from $\mathcal{A}$. It chooses $\beta \in \{0,1\}$ at random. To encrypt M_β, $\mathcal{B}$ chooses $\mathsf{ctag}^*_1, \ldots, \mathsf{ctag}^*_\ell$ at random from $\mathbb{Z}_p$ and computes the ciphertext elements as follows.

$$C_0 = M_\beta \cdot e(sP_1, P_2)^\alpha$$

$$C_1 = sP_1,\ C_2 = Z_1,\ C_3 = -y_v(sP_1) - y'_v Z_1 + y_w(sP_1)$$

$$E_j = (\mathsf{id}^*_j y_{q_j} + \mathsf{ctag}^*_j y_w + y_{u_j})(sP_1) \text{ for } 1 \le j \le \ell$$

$\mathcal{B}$ returns $\mathcal{C}^* = (C_0, C_1, C_2, C_3, E_1, \ldots, E_\ell, \mathsf{ctag}^*_1, \ldots, \mathsf{ctag}^*_\ell)$ to $\mathcal{A}$.

If $Z_1 = asP_1$ then the challenge ciphertext is normal; otherwise if Z_1 is a random element of $\mathbb{G}_1$ i.e., $Z_1 = (as + c)P_1$ then the ciphertext is semi-functional with $\mu = c$.

Note that, to check whether $\mathcal{C}^*$ is semi-functional or not, $\mathcal{B}$ itself could try to decrypt it with a semi-functional key for $\overrightarrow{\mathsf{id}}^*$. However since aP_2 is not known to $\mathcal{B}$, it cannot create such a key.

Phase 2: As in first phase, $\mathcal{B}$ returns a normal key for every query.

Guess: The adversary returns its guess β' to $\mathcal{B}$.

If $\mathcal{C}^*$ is normal then $\mathcal{B}$ simulates Game$_{real}$ and when it is semi-functional, $\mathcal{B}$ simulates Game$_0$. Therefore if $\mathcal{A}$ is able to distinguish between Game$_{real}$ and Game$_0$ i.e., if $\beta = \beta'$, then the simulator can solve the DDH1 problem with advantage

$$\mathsf{Adv}^{\mathcal{B}}_{\text{DDH1}} = |\Pr[X_{real}] - \Pr[X_0]| = \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{real}} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_0} = \varepsilon$$

$\square$

Lemma 8. *If there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_k} = \varepsilon$, then we can build an algorithm $\mathcal{B}$ having advantage $q_A\varepsilon$ in breaking the assumption* DDH2v.

Proof. Let $(P_1, dP_1, dzP_1, zx_1P_1, P_2, dP_2, x_1P_1, x_2P_2, Z_2)$ be the instance of DDH2v that $\mathcal{B}$ receives.

Setup: $\mathcal{B}$ chooses random elements $a, \alpha, \lambda_1, \ldots, \lambda_n, \nu_1, \ldots, \nu_n, y'_v, y_{q_1}, \ldots, y_{q_n}, y_{u_1}, \ldots, y_{u_n}, y_w \in_R \mathbb{Z}_p$ and sets the parameters as follows. $P_1 = P_1, P_2 = P_2$, $Q_{2,j} = -\lambda_j(dP_2) + y_{q_j}P_2$, $U_{2,j} = -\nu_j(dP_2) + y_{u_j}P_2$ for $1 \leq j \leq n$, $W_2 = dP_2 + y_wP_2$, $V_2 = -a(x_1P_2)$ and $V'_2 = x_1P_2 + y'_vP_2$ setting $\tau = ay'_v$ using which one can compute $\tau P_1 = ay'_vP_1$. The public parameters $Q_{1,1}, \ldots, Q_{1,n}, W_1, U_{1,1}, \ldots, U_{1,n}$ can be computed since $\mathcal{B}$ has dP_1. The remaining parameters required to provide $\mathcal{PP}$ to $\mathcal{A}$ are computed using a, α and other elements of the problem instance.

Phases 1 and 2: We first describe how create, delegate and reveal queries are handled. The simulator $\mathcal{B}$ chooses $\theta \in \{1, 2, \ldots, q_A\}$ at random and guesses that the k-th key revealed will be the θ-th key either created directly or by delegation. It then creates a counter $\jmath$ for the number of create/delegate queries and initializes it to zero. $\mathcal{B}$ now considers two cases.

Case $\jmath \neq \theta$: If this is a create query for an identity $\overrightarrow{\mathsf{id}}$ of depth ℓ then $\mathcal{B}$ chooses tags $\mathsf{ktag}_1, \ldots, \mathsf{ktag}_\ell$ at random and associates them along with $\overrightarrow{\mathsf{id}}$ to the $\jmath$-th member of the set S. Otherwise, if this is a delegate query with inputs $\overrightarrow{\mathsf{id}}$ of depth $\ell - 1$ and an identity id_ℓ, then the simulator chooses one new tag ktag_ℓ. The other $\ell - 1$ tags $\mathsf{ktag}_1, \ldots, \mathsf{ktag}_{\ell-1}$ are copied from the key we are delegating from. All these tags are associated with the $\jmath$-th element of S.

Case $\jmath = \theta$: If this is a create query for an identity $\overrightarrow{\mathsf{id}} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ of depth ℓ then $\mathcal{B}$ chooses tags $\mathsf{ktag}_1, \ldots, \mathsf{ktag}_{\ell-1}$ at random, sets $\mathsf{ktag}_\ell = \lambda_\ell \mathsf{id}_\ell + \nu_\ell$ and associates these tags along with $\overrightarrow{\mathsf{id}}$ to the $\jmath$-th member of the set S. Otherwise, if this is a delegate query with inputs $\overrightarrow{\mathsf{id}}$ of depth $\ell - 1$ and an identity id_ℓ, then the simulator sets $\mathsf{ktag}_\ell = \lambda_\ell \mathsf{id}_\ell + \nu_\ell$. The other $\ell - 1$ tags $\mathsf{ktag}_1, \ldots, \mathsf{ktag}_{\ell-1}$ are copied from the key we are delegating from. $\mathcal{B}$ associates these tags with the $\jmath$-th element of S.

Each element of the set S is now associated with tag values but not a key. This is sufficient since the only elements that are not re-randomized during delegation are the tags. So the keys can be constructed just before revealing. Now consider the $\imath$-th reveal query for $\imath \in \{1, \ldots, q_R\}$ and suppose that this query is for the $\jmath$-th key to be revealed. For $\imath > k$ the simulator has to create a normal key and does so using the master secret and the tag values stored in the $\jmath$-th element of S. For $\imath < k$, $\mathcal{B}$ creates a semi-functional key by first creating a normal key using the tag values stored in the $\jmath$-th element of S and then modifies it using its knowledge of aP_2.

Now consider the case when $\imath = k$. If $\jmath \neq k$ then the simulator aborts and makes a random guess of the distribution of Z_2. Otherwise, using the master secret, it generates a normal key $\mathcal{SK}'_{\overrightarrow{\mathsf{id}}}$ for the identity $\overrightarrow{\mathsf{id}} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$ with components $K'_1, K'_2, K'_{3,1}, \ldots, K'_{3,\ell}, D'_1, \ldots, D'_\ell$ and tag values already assigned. We know that $\mathsf{ktag}_\ell = \lambda_\ell \mathsf{id}_\ell + \nu_\ell$. Let $r'_1, \ldots, r'_\ell$ be the randomisers used with $r' = r'_1 + \cdots + r'_\ell$. $\mathcal{B}$ then chooses $\gamma \in \mathbb{Z}_p$ at random, sets

$$K_1 = K'_1 - aZ_2,\ K_2 = K'_2 + Z_2 + y'_v(x_2P_2)$$

$$K_{3,j} = K'_{3,j},\ D_j = D'_j \text{ for } 1 \leq j \leq \ell - 1$$

$$K_{3,\ell} = K'_{3,\ell} + x_2P_2$$

$$D_\ell = D'_\ell + (y_{q_\ell}\mathsf{id}_\ell + y_w\mathsf{ktag}_\ell + y_{u_\ell})(x_2P_2)$$

thus implicitly setting $r_\ell = r'_\ell + x_2$ and $r = r' + x_2$. It returns the key $\mathcal{SK}_{\overrightarrow{\mathsf{id}}}$ consisting of $K_1, K_2, K_{3,1}, \ldots, K_{3,\ell}, D_1, \ldots, D_\ell, \mathsf{ktag}_1, \ldots, \mathsf{ktag}_\ell$ to $\mathcal{A}$. The choice of $\mathsf{ktag}_\ell = \lambda_\ell\mathsf{id}_\ell + \nu_\ell$ allows $\mathcal{B}$ to create D_ℓ. If $Z_2 = x_1x_2P_2$ then the key for $\overrightarrow{\mathsf{id}}$ will be normal and otherwise it will be semi-functional with $\gamma = c$ where $Z_2 = (x_1x_2 + c)P_2$. Note that a semi-functional ciphertext for $\overrightarrow{\mathsf{id}}$ with any value of ctag_ℓ cannot be generated without the knowledge of dx_1zP_1 which is neither available from the assumption nor can be computed by $\mathcal{B}$. This rules out the obvious way of checking whether the key for id_ℓ is semi-functional or not.

Challenge: $\mathcal{B}$ receives two messages M_0, M_1 and a challenge identity $\overrightarrow{\mathsf{id}}^* = (\mathsf{id}^*_1, \ldots, \mathsf{id}^*_\ell)$ during the challenge phase. It chooses $\beta \in_R \{0,1\}$, generates a normal ciphertext with randomisers $s' \in_R \mathbb{Z}_p$ and $\mathsf{ctag}^*_j = \lambda_j\mathsf{id}^*_j + \nu_j$ for $1 \leq j \leq \ell$. Observe that the tags are chosen this way to be able to create the ciphertext elements $E_1, \ldots, E_\ell$. The simulator then changes the ciphertext elements as follows.

$$C_0 = C'_0 \cdot e(zx_1P_1, P_2)$$

$$C_1 = C'_1 + x_1zP_1,\ C_2 = C'_2 + a(zx_1P_1) + dzP_1,\ C_3 = C'_3 - ay'_v(zx_1P_1) + y_w(zx_1P_1) - y'_v(dzP_1)$$

$$E_j = E'_j + (y_{q_j}\mathsf{id}_j + \mathsf{ctag}^*_j y_w + y_{u_j})(zx_1P_1) \text{ for } 1 \leq j \leq \ell$$

setting $s = s' + zx_1$ and $\mu = dz$.

Now $\mathcal{A}$ will be able to distinguish between Game$_{k-1}$ and Game$_k$ if it can decide whether $\mathcal{SK}_{\overrightarrow{\mathsf{id}}}$ is normal or semi-functional. When the game $\jmath \neq k$ then the guess is random and the probability that $\mathcal{A}$ wins is 1/2. So we need only consider the event $\jmath = k$ which happens with probability $1/q_A$. Then $\mathcal{B}$ can break the assumption DDH2v with advantage

$$\mathsf{Adv}^{\mathcal{B}}_{\text{DDH2v}} = |\Pr[X_{k-1}] - \Pr[X_k]| = q_A(\mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_k}) = q_A\varepsilon$$

□

Lemma 9. *If there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{q_R}} - \mathsf{Adv}^{\mathcal{A}}_{final} = \varepsilon$, then we can build an algorithm $\mathcal{B}$ having advantage ε in breaking the* DBDH *assumption.*

Proof. $\mathcal{B}$ receives $(P_1, xP_1, aP_1, sP_1, P_2, xP_2, aP_2, sP_2, Z)$ as an instance of the DBDH problem.

Setup: With $b, y_v, y'_v, y_{q_1}, \ldots, y_{q_n}, y_w, y_{u_1}, \ldots, y_{u_n}$ chosen at random from $\mathbb{Z}_p$, $\mathcal{B}$ sets the parameters as

$$P_1 = P_1, P_2 = P_2, aP_1 = aP_1, V_2 = y_v P_2, V'_2 = y'_v P_2,\ \tau P_1 = y_v P_1 + y'_v(aP_1)$$
$$Q_{1,j} = y_{q_j} P_1, W_1 = y_w P_1, U_{1,j} = y_{u_j} P_1 \text{ for } 1 \le j \le n, e(P_1, P_2)^{b\alpha} = e(xP_1, aP_2)$$

implicitly setting $a = a$, $\alpha = xa$ and $\tau = y_v + ay'_v$. The remaining parameters can be computed easily. $\mathcal{B}$ returns $\mathcal{PP}$ to $\mathcal{A}$.

Phases 1 and 2: When $\mathcal{A}$ asks for the secret key for an identity vector $\overrightarrow{\mathsf{id}} = (\mathsf{id}_1, \ldots, \mathsf{id}_\ell)$, $\mathcal{B}$ chooses at random $r_1, \ldots, r_\ell, \mathsf{ktag}_1, \ldots, \mathsf{ktag}_\ell, \gamma' \in \mathbb{Z}_p$ with $r = r_1 + \cdots + r_\ell$ implicitly setting $\gamma' = x - \gamma$. It then computes a semi-functional key for $\overrightarrow{\mathsf{id}}$ as follows.

$$K_1 = \gamma'(aP_2) + rV_2 = xaP_2 - a\gamma P_2 + rV_2 = \alpha P_2 + rV_2 - a\gamma P_2$$
$$K_2 = rV'_2 - \gamma' P_2 = rV'_2 + xP_2 + \gamma P_2 = rV'_2 + \gamma P_2$$
$$K_{3,j} = r_j P_2, D_j = r_j(\mathsf{id}_j Q_{2,j} + \mathsf{ktag}_j W_2 + U_{2,j}) \text{ for } 1 \le j \le \ell.$$

Observe that $\mathcal{B}$ does not know α and hence cannot create a normal key.

Challenge: $\mathcal{B}$ receives the challenge identity $\overrightarrow{\mathsf{id}}^* = (\mathsf{id}^*_1, \ldots, \mathsf{id}^*_\ell)$ and two messages M_0 and M_1 from $\mathcal{A}$. It chooses $\beta \in \{0,1\}$ and $\mathsf{ctag}_1*, \ldots, \mathsf{ctag}^*_\ell, \mu', t \in \mathbb{Z}_p$ at random and generates a semi-functional challenge ciphertext as follows. Here $\mathcal{B}$ implicitly sets $\mu' = \mu + as$.

$$C_0 = M_\beta \cdot Z^b,$$
$$C_1 = sP_1, C_2 = \mu' P_1 = asP_1 + \mu P_1$$
$$C_3 = -y_v(sP_1) - \mu' y'_v P_1 + y_w(sP_1) = -y_v sP_1 - asy'_v P_1 - \mu y'_v P_1 + sW_1 = -\tau s P_1 - \mu V'_1 + sW_1$$
$$E_j = (y_{q_j}\mathsf{id}^*_j + y_w \mathsf{ctag}^*_j + y_{u_j})(sP_1) \text{ for } 1 \le j \le \ell$$

The challenge ciphertext $\mathcal{C}^* = (C_0, C_1, C_2, C_3, E_1, \ldots, E_\ell, \mathsf{ctag}^*_1, \ldots, \mathsf{ctag}^*_\ell)$ is returned to $\mathcal{A}$. If $Z = e(P_1, P_2)^{xas}$ then $\mathcal{C}^*$ will be a semi-functional encryption of M_β; if Z is a random element of $\mathbb{G}_T$ then $\mathcal{C}^*$ will be a semi-functional encryption of a random message. If $\mathcal{A}$ can identify whether the game simulated was Game$_q$ or Game$_{final}$, then $\mathcal{B}$ will be able to decide whether $Z = e(P_1, P_2)^{xas}$ or not and hence break the DBDH assumption with advantage

$$\mathsf{Adv}^{\mathcal{B}}_{\mathrm{DBDH}} = |\Pr[X_{q_R}] - \Pr[X_{final}]| = \mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_{q_R}} - \mathsf{Adv}^{\mathcal{A}}_{\mathrm{Game}_{final}} = \varepsilon$$

□

Theorem 4. *If the* DDH1, DDH2v *and* DBDH *assumptions hold, then no polynomial time adversary $\mathcal{A}$ making at most q_R reveal queries can break the security of the HIBE scheme.*

Proof. Using lemmas 7, 8 and 9, we have for any polynomial time attacker $\mathcal{A}$,

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}}_{\mathrm{HIBE}} &\le |\Pr[X_{real}] - \Pr[X_0]| + \sum_{k=1}^{q_R}(|\Pr[X_{k-1}] - \Pr[X_k]|) + |\Pr[X_{q_R}] - \Pr[X_{final}]| \\ &= \varepsilon_{\mathrm{DDH1}} + q_R q_A \times \varepsilon_{\mathrm{DDH2v}} + \varepsilon_{\mathrm{DBDH}}\end{aligned}$$

which is negligible in the security parameter κ. □

H Security Proof for the BE Scheme in Section 4.2

The proof is organised as a sequence of games. Game_{real} denotes the real BE security game as defined in Section A.3. Game_0 is similar to Game_{real} except that the ciphertext encrypted to the challenge set is semi-functional. Suppose that the adversary queries for private keys of q users. In Game_k for $1 \leq k \leq q$, the private keys returned for the first k queries are semi-functional and the rest are normal. Game_{final} is just like Game_q except that the challenge ciphertext is an encryption of a random message. Define X_{real}, X_k for $0 \leq k \leq q$ and X_{final} as earlier.

Lemma 10. *If there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{real}} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_0} = \varepsilon$, then we can build an algorithm $\mathcal{B}$ having advantage ε in solving the* DDH1 *problem.*

Proof. The algorithm $\mathcal{B}$ receives $(P_1, sP_1, aP_1, P_2, Z_1)$ as an instance of DDH1. We describe how it will simulate each phase in the security game.

Setup: $\mathcal{B}$ chooses random elements $\alpha, y_v, y'_v, y_{q_1}, \ldots, y_{q_n}, y_w$ from $\mathbb{Z}_p$ and sets the parameters as follows.

$$P_1 = P_1, sP_1 = sP_1, aP_1 = aP_1, Q_{1,i} = y_{q_i}P_1 \text{ for } 1 \leq i \leq n, W_1 = y_w P_1$$
$$P_2 = P_2, V_2 = y_v P_2, V'_2 = y'_v P_2$$

This implicitly sets $\tau = y_v + ay'_v$. Using this, the element τP_1 can be computed as $y_v P_1 + y'_v(aP_1)$. The simulator computes the remaining parameters using α and gives the public key $\mathcal{PK}$ to $\mathcal{A}$.

Private Key Queries: $\mathcal{A}$ issues a number of private key queries adaptively. $\mathcal{B}$ returns a normal key for every key query computed using $\mathcal{SK}$.

Challenge: $\mathcal{B}$ receives the challenge set S^* and two messages M_0 and M_1 from $\mathcal{A}$. It chooses $\beta \in \{0, 1\}$ at random and computes the ciphertext elements as follows.

$$C_0 = M_\beta \cdot e(sP_1, P_2)^\alpha$$
$$C_1 = sP_1,\ C_2 = Z_1,\ C_3 = -y_v(sP_1) - y'_v Z_1 + y_w(sP_1)$$
$$E = (\sum_{i \in S^*} y_{q_i})(sP_1)$$

$\mathcal{B}$ returns $\mathcal{C}^* = (C_0, C_1, C_2, C_3, E)$ to $\mathcal{A}$. If $Z_1 = asP_1$ then the challenge ciphertext is normal; otherwise if Z_1 is a random element of $\mathbb{G}_1$ i.e., $Z_1 = (as+c)P_1$ then the ciphertext is semi-functional with $\mu = c$.

Note that, to check whether $\mathcal{C}^*$ is semi-functional or not, $\mathcal{B}$ itself could try to decrypt it with a semi-functional key for some user $i \in S^*$. However since aP_2 is not known to $\mathcal{B}$, it cannot create such a key.

Guess: The adversary returns its guess β' to $\mathcal{B}$.

If $\mathcal{C}^*$ is normal then $\mathcal{B}$ simulates Game_{real} and it is semi-functional $\mathcal{B}$ simulates Game_0. Therefore if $\mathcal{A}$ is able to distinguish between Game_{real} and Game_0 i.e., if $\beta = \beta'$, then the simulator can solve the DDH1 problem with advantage

$$\mathsf{Adv}^{\mathcal{B}}_{\text{DDH1}} = |\Pr[X_{real}] - \Pr[X_0]| = \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{real}} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_0} = \varepsilon$$

□

Lemma 11. *If there exists an adversary $\mathcal{A}$ making at most q private key queries such that* $\mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_k} = \varepsilon$*, then we can build an algorithm $\mathcal{B}$ having advantage at most $q\varepsilon$ in breaking the assumption* DDH2v.

Proof. Let $(P_1, dP_1, dzP_1, zx_1P_1, P_2, dP_2, x_1P_1, x_2P_2, Z_2)$ be the instance of DDH2v that $\mathcal{B}$ receives.

Setup: $\mathcal{B}$ chooses random elements $a, \alpha, y'_v, y_{q_1}, \ldots, y_{q_n}, y_w \in_R \mathbb{Z}_p$ and sets the parameters as follows. $\mathcal{B}$ guesses a value $j' \in \{1, \ldots, q\}$. $P_1 = P_1, P_2 = P_2$, $Q_{2,j'} = -dP_2 + y_{q_{j'}}P_2$, $Q_{2,i} = y_{q_i}P_2$ for $i \neq j'$, $W_2 = dP_2 + y_wP_2$, $V_2 = -a(x_1P_2)$ and $V'_2 = x_1P_2 + y'_vP_2$ setting $\tau = ay'_v$ using which one can compute $\tau P_1 = ay'_vP_1$. The public parameters $Q_{1,1}, \ldots, Q_{1,n}, W_1$ can be computed since $\mathcal{B}$ has dP_1. The remaining parameters required to provide $\mathcal{PK}$ to $\mathcal{A}$ are computed using a, α and other elements of the problem instance.

Private Key Queries: The queries before the k-th query are answered with a semi-functional key and for those after k-th query, normal keys are returned. Suppose that the k-th query is for the private key of user j.

When $j \neq j'$, the simulator $\mathcal{B}$ returns a normal key to $\mathcal{A}$ and randomly guesses whether $Z_2 = x_1x_2P_2$ or not. In such a case, the adversary has zero advantage in distinguishing between the games. Also $\mathcal{B}$ has zero advantage in solving the DDH2v problem.

Next we consider the case $j = j'$ which happens with probability at least $1/q$. For the k-th query (for user j) a normal key $\mathcal{SK}_j$ with elements $K'_1, K'_2, K'_3, D, \forall_{i \neq j} D_i$ is generated using randomiser $r' \in_R \mathbb{Z}_p$ and then modified as:

$$K_1 = K'_1 - aZ_2,\ K_2 = K'_2 + Z_2 + y'_v(x_2P_2),\ K_3 = K'_3 + x_2P_2$$

$$D = D' + (y_{q_j} + y_w)(x_2P_2),\ D_i = D'_i + y_{q_i}(x_2P_2) \text{ for all } i \neq j.$$

Since $j = j'$,

$$\begin{aligned} D &= D' + (y_{q_j} + y_w)(x_2P_2) \\ &= r'(Q_{2,j} + W_2) + x_2(y_{q_j} + y_w)P_2 \\ &= r'(-dP_2 + y_{q_j}P_2 + dP_2 + y_wP_2) + x_2(y_{q_j} + y_w)P_2 \\ &= (r' + x_2)(y_{q_j} + y_w)P_2 \\ &= (r' + x_2)(-d + y_{q_j} + d + y_w)P_2 \\ &= (r' + x_2)(Q_{2,j} + W_2). \end{aligned}$$

This implicitly sets $r = r' + x_2$. If $Z_2 = x_1x_2P_2$ then the key $\mathcal{SK}_j$ will be normal and otherwise it will be semi-functional with $\gamma = c$ where $Z_2 = (x_1x_2 + c)P_2$. Note that a semi-functional ciphertext for any set containing user j cannot be generated without V'_1 and thus the obvious way of checking whether $\mathcal{SK}_j$ is semi-functional or not is ruled out.

Challenge: $\mathcal{B}$ receives two messages M_0, M_1 and a challenge set S^*. It chooses $\beta \in_R \{0, 1\}$, generates a normal ciphertext with elements $C'_0, C'_1, C'_2, C'_3, E'$ generated using the randomiser $s' \in_R \mathbb{Z}_p$ and changes the ciphertext elements as follows.

$$C_0 = C'_0 \cdot e(zx_1P_1, P_2)$$

$$C_1 = C_1' + x_1zP_1,\ C_2 = C_2' + a(zx_1P_1) + dzP_1,\ C_3 = C_3' - ay_v'(zx_1P_1) + y_w(zx_1P_1) - y_v'(dzP_1)$$

$$E = E' + (\sum_{i \in S^*} y_{q_i})(zx_1P_1)$$

setting $s = s' + zx_1$ and $\mu = dz$. It is easy to check that C_3 is well-formed.

Now $\mathcal{A}$ will be able to distinguish between Game$_{k-1}$ and Game$_k$ if it can decide whether $\mathcal{SK}_j$ is normal or semi-functional. If it succeeds $\mathcal{B}$ can break the assumption DDH2v with advantage

$$\mathsf{Adv}^{\mathcal{B}}_{\text{DDH2v}} = |\Pr[X_{k-1}] - \Pr[X_k]| \leq q(\mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{k-1}} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_k}) = q\varepsilon$$

□

Lemma 12. *If there exists an adversary $\mathcal{A}$ such that $\mathsf{Adv}^{\mathcal{A}}_{\text{Game}_q} - \mathsf{Adv}^{\mathcal{A}}_{final} = \varepsilon$, then we can build an algorithm $\mathcal{B}$ having advantage ε in breaking the* DBDH *assumption.*

Proof. $\mathcal{B}$ receives $(P_1, xP_1, aP_1, sP_1, P_2, xP_2, aP_2, sP_2, Z)$ as an instance of the DBDH problem.

Setup: With $b, y_v, y_v', y_{q_1}, \ldots, y_{q_n}, y_w$ chosen at random from $\mathbb{Z}_p$, $\mathcal{B}$ sets the parameters as

$$P_1 = P_1, P_2 = P_2, aP_1 = aP_1, V_2 = y_vP_2, V_2' = y_v'P_2,\ \tau P_1 = y_vP_1 + y_v'(aP_1)$$

$$Q_{1,1} = y_{q_1}P_1, \ldots, Q_{1,n} = y_{q_n}P_1, W_1 = y_wP_1, e(P_1, P_2)^{b\alpha} = e(xP_1, aP_2)$$

implicitly setting $a = a$, $\alpha = xa$ and $\tau = y_v + ay_v'$. The remaining parameters can be computed easily. $\mathcal{B}$ returns $\mathcal{PP}$ to $\mathcal{A}$.

Private Key Queries: When $\mathcal{A}$ asks for the secret key for user j, $\mathcal{B}$ chooses at random $r, \gamma' \in \mathbb{Z}_p$ implicitly setting $\gamma' = x - \gamma$. It then computes a semi-functional key for j as follows.

$$K_1 = \gamma'(aP_2) + rV_2 = xaP_2 - a\gamma P_2 + rV_2 = \alpha P_2 + rV_2 - a\gamma P_2$$

$$K_2 = rV_2' - \gamma' P_2 = rV_2' + xP_2 + \gamma P_2 = rV_2' + \gamma P_2$$

$$K_3 = rP_2,\ D = r(Q_{2,j} + W_2),\ D_i = rQ_{2,i} \text{ for } i \neq j.$$

Observe that $\mathcal{B}$ does not know α and hence cannot create a normal key.

Challenge: $\mathcal{B}$ receives the challenge set S^* and two messages M_0 and M_1 from $\mathcal{A}$. It chooses $\beta \in \{0, 1\}$ and $\mu' \in \mathbb{Z}_p$ at random and generates a semi-functional challenge ciphertext as follows. Here $\mathcal{B}$ implicitly sets $\mu' = \mu + as$.

$$C_0 = M_\beta \cdot Z^b,$$

$$C_1 = sP_1, C_2 = \mu' P_1 = asP_1 + \mu P_1$$

$$C_3 = -y_v(sP_1) - \mu' y_v' P_1 + y_w sP_1 = -y_v sP_1 - asy_v'P_1 - \mu y_v' P_1 + y_w sP_1 = -\tau sP_1 - \mu V_1' + sW_1$$

$$E = (\sum_{i \in S^*} y_{q,i})(sP_1)$$

The challenge ciphertext $\mathcal{C}^* = (C_0, C_1, C_2, C_3, E, \mathsf{ctag}^*)$ is returned to $\mathcal{A}$. If $Z = e(P_1, P_2)^{xas}$ then $\mathcal{C}^*$ will be a semi-functional encryption of M_β to S^*; if Z is a random element of $\mathbb{G}_T$ then $\mathcal{C}^*$ will be a semi-functional encryption of a random message. If $\mathcal{A}$ can identify whether the game simulated was Game$_q$ or Game$_{final}$, then $\mathcal{B}$ will be able to decide whether $Z = e(P_1, P_2)^{xas}$ or not and hence break the DBDH assumption with advantage

$$\mathsf{Adv}^{\mathcal{B}}_{\text{DBDH}} = |\Pr[X_q] - \Pr[X_{final}]| = \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_q} - \mathsf{Adv}^{\mathcal{A}}_{\text{Game}_{final}} = \varepsilon$$

□

Theorem 5. *If the* DDH1, DDH2v *and* DBDH *assumptions hold, then no polynomial time adversary $\mathcal{A}$ making at most q private key queries can break the security of the broadcast encryption scheme.*

Proof. Using lemmas 10, 11 and 12, we have for any polynomial time attacker $\mathcal{A}$,

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}}_{\text{BE}} &\leq |\Pr[X_{real}] - \Pr[X_0]| + \sum_{k=1}^{q} (|\Pr[X_{k-1}] - \Pr[X_k]|) + |\Pr[X_q] - \Pr[X_{final}]| \\ &\leq \varepsilon_{\text{DDH1}} + q^2 \varepsilon_{\text{DDH2v}} + \varepsilon_{\text{DBDH}}\end{aligned}$$

which is negligible in the security parameter κ. □