The BBG HIBE Has Limited Delegation

Hovav Shacham*
hovav.shacham@weizmann.ac.il

Abstract

At Eurocrypt 2005, Boneh, Boyen, and Goh presented a hierarchical IBE for which they claimed a novel property, called limited delegation: it is possible to give an entity a private key that restricts it from generating descendant private keys beyond some depth d; in particular, with d equal to the entity's depth, such a key allows decryption only. In this paper, we argue that this claim is nonobvious and requires proof, provide a precise model for arguing about limited delegation, and prove that the Boneh-Boyen-Goh system does, in fact, have limited delegation. Whereas Boneh, Boyen, and Goh prove their system semantically secure under the BDHI assumption, our proof of limited delegation requires the stronger BDHE assumption.

1 Introduction

At Eurocrypt 2005, Boneh, Boyen, and Goh presented a hierarchical IBE system ("BBG") [2] in which ciphertexts are three group elements regardless of the depth in the hierarchy of the identity encrypted to. In addition, they make the following claim: "an identity ID at depth k can be given a private key that only lets it issue private keys to descendants of bounded depth." The BBG HIBE is the first HIBE with this property, called "limited delegation."

The purpose of this paper is threefold. First, we argue that the claim that BBG has limited delegation is nonobvious and requires proof. Second, we provide a precise model for arguing about limited delegation. Third, we prove that the BBG system does, in fact, have limited delegation.

Perhaps surprisingly, our proof makes use of the bilinear Diffie-Hellman exponent (BDHE) assumption, rather than the weaker bilinear Diffie-Hellman inversion (BDHI) assumption. The proceedings version of [2] introduced the BDHE assumption to prove the security of the BBG scheme, and BDHE was later also used to construct an efficient public-key broadcast encryption scheme [3]. However, it was later observed (by Nelly Fazio) that the proof requires only BDHI, a weaker assumption previously used in an IBE construction [1].

2 Mathematical Background

We recall the pairing-based crypto setting (see also [4, 5]) and the BDHI and BDHE assumptions. Consider:

- G and G_1 are multiplicative cyclic groups of order p;
- the group action on G and G_1 can be computed efficiently;

*Supported by a Koshland Scholars Program fellowship.

- g is a generator of G;
- $e\colon G \times G \to G_1$ is an efficiently computable map with the following properties:
  - Bilinear: for all $u, v \in G$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$;
  - Non-degenerate: $e(g, g) \neq 1$.

We say that G is a bilinear group if it satisfies these requirements.

In our descriptions of the assumptions, g and h are random elements of G and α is a random element of $\mathbb{Z}_p$.

The computational ℓ-bilinear Diffie-Hellman inversion (BDHI) problem is as follows:

$$\ell\text{-BDHI:} \qquad \text{given} \quad g, g^{\alpha}, g^{\alpha^2}, g^{\alpha^3}, \ldots, g^{\alpha^{\ell}} \quad \text{compute} \quad e(g,g)^{1/\alpha} \ .$$

Boneh, Boyen, and Goh use a somewhat weaker variant called ℓ-wBDHI*, which is as follows:

$$\ell\text{-wBDHI*:} \qquad \text{given} \quad g, h, g^{\alpha}, g^{\alpha^2}, g^{\alpha^3}, \ldots, g^{\alpha^{\ell}} \quad \text{compute} \quad e(g,h)^{(\alpha^{\ell+1})} \ .$$

Finally, the ℓ-bilinear Diffie-Hellman exponent (BDHE) problem is as follows:

$$\ell\text{-BDHE:} \qquad \text{given} \quad g, h, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^{\ell-1}}, g^{\alpha^{\ell+1}}, g^{\alpha^{\ell+2}}, \ldots, g^{\alpha^{2\ell}}, \quad \text{compute} \quad e(g,h)^{(\alpha^{\ell+1})} \ .$$

Each of these problems has a decisional variant in which the adversary is also given a value T that is either the solution to the corresponding computational problem or a random element of G_1. The computational and decisional assumptions are formalized in the standard way; see Boneh, Boyen, and Goh [2] for more details. In what follows, when we refer to the BDHI or BDHE assumption, we are in fact referring to the decisional variant of wBDHI* or BDHE.

3 The BBG System

We recall the BBG system for an ℓ-level hierarchy. See [2] for motivation and details. For the reader's convenience, we retain the notation of Boneh, Boyen, and Goh exactly.

`Setup`(ℓ)**:** Choose $g \xleftarrow{R} G$ and $\alpha \xleftarrow{R} \mathbb{Z}_p$ and set $g_1 \leftarrow g^{\alpha}$. Now, choose $g_2, g_3, h_1, \ldots, h_{\ell} \xleftarrow{R} G$. Set:

$$\textit{params} \leftarrow (g_1, g_2, g_3, h_1, \ldots, h_{\ell}) \qquad \text{and} \qquad \textit{master-key} \leftarrow g_2^{\alpha} \ .$$

`KeyGen`($d_{\mathsf{ID}|k-1}, \mathsf{ID}$)**:** Parse ID as $(I_1, \ldots, I_k) \in (\mathbb{Z}_p^*)^k$ with depth $k \leq \ell$. To generate d_{ID} directly using the master secret, choose $r \xleftarrow{R} \mathbb{Z}_p$ and compute and output

$$d_{\mathsf{ID}} \leftarrow \left(g_2^{\alpha} \cdot \left(h_1^{I_1} \cdots h_k^{I_k} \cdot g_3\right)^r,\ g^r,\ h_{k+1}^r, \ldots, h_{\ell}^r \right) \ .$$

To generate the private key using the key $d_{\mathsf{ID}|k-1}$ for parent identity $\mathsf{ID}|k-1 = (I_1, \ldots, I_{k-1})$ instead, parse $d_{\mathsf{ID}|k-1} = (a_0, a_1, b_k, \ldots, b_{\ell})$; choose $t \xleftarrow{R} \mathbb{Z}_p$ and set

$$d_{\mathsf{ID}} \leftarrow \left(a_0 \cdot b_k^{I_k} \cdot \left(h_1^{I_1} \cdots h_k^{I_k} \cdot g_3\right)^t,\ a_1 \cdot g^t,\ b_{k+1} \cdot h_{k+1}^t, \ldots, b_{\ell} \cdot h_{\ell}^t \right) \ .$$

`Encrypt`$(params, \mathsf{ID}, M)$**:** To encrypt $M \in G_1$ to identity $\mathsf{ID} = (I_1, \ldots, I_k) \in (\mathbb{Z}_p^*)^k$, choose $s \stackrel{\mathrm{R}}{\leftarrow} \mathbb{Z}_p$ and compute and output

$$\mathsf{CT} \leftarrow \left(e(g_1, g_2)^s \cdot M,\ g^s,\ \left(h_1^{I_1} \cdots h_k^{I_k} \cdot g_3 \right)^s \right) \ .$$

`Decrypt`$(d_{\mathsf{ID}}, \mathsf{CT})$**:** To decrypt the ciphertext $\mathsf{CT} = (A, B, C)$ using the private key $d_{\mathsf{ID}} = (a_0, a_1, b_{k+1}, \ldots, b_\ell)$ for the identity $\mathsf{ID} = (I_1, \ldots, I_k)$, compute and output

$$M \leftarrow A \cdot e(a_1, C) \ / \ e(B, a_0) \ .$$

4 Limited Delegation

Consider a BBG private key for a level k identity, of the form $(a_0, a_1, b_{k+1}, \ldots, b_\ell)$. We make some observations:

- The decryption algorithm makes use only of the first two components a_0 and a_1.
- In generating the private key for a level-$(k+1)$ child identity, the key generation algorithm requires only b_{k+1} in addition to a_0 and a_1; the remaining values $(b_{k+1}, \ldots, b_\ell)$ are rerandomized and passed down to the child key.

An entity that is given only the restricted private key (a_0, a_1), then, can run `Decrypt` but not `KeyGen`; and, more generally, an entity that is given the restricted private key $(a_0, a_1, b_{k+1}, \ldots, b_d)$ for some $k < d < \ell$ can run `KeyGen` to produce private keys for its descendants, but only up to depth d. This is what Boneh, Boyen, and Goh call limited delegation.

It does not necessarily follow, however, that because it is not possible to use `KeyGen` to derive private keys beyond some depth it is impossible to derive these keys by some other means. And even if it is impossible to derive these keys, it might still be possible to derive enough information about them to distinguish ciphertexts encrypted to their corresponding identities. To show that the BBG scheme has limited delegation, we need to give a proof that rules out attacks of this sort.

To begin with, of course, we must give a precise definition for what limited delegation is. We now give two such definitions, one for the adaptive-ID setting, a second for selective-ID setting. (We give a separate selective-ID definition because, as we will see, the selective-ID setting allows a substantial simplification in the definition.)

Before, we note that the depth-restriction feature of BBG is actually a fifth algorithm, behaving as follows:

`Restrict`(d_{ID}, d)**:** Here d_{ID} is a (possibly restricted) private key for a depth-k identity, and $d \geq k$ is the depth to which it should be restricted. Parse $d_{\mathsf{ID}} = (a_0, a_1, b_{k+1}, \ldots, b_{d'})$, where d' is the depth to which d_{ID} is presently restricted. For unrestricted keys, $d' = \ell$. If d is greater than d', return an error condition, since a key cannot be unrestricted. If k equals d, remove all b-elements, returning the newly restricted private key (a_0, a_1). Otherwise, remove the elements $b_{d+1}, \ldots, b_{d'}$, returning the newly restricted private key $(a_0, a_1, b_{k+1}, \ldots b_d)$.

4.1 The Restricted-Delegation Game for Full HIBE

In the standard IND-ID-CPA game, the adversary is allowed to make private key extraction queries for arbitrary IDs. To model restricted keys, we must also allow the adversary to make restricted-key queries, in which he specifies the depth d up to which he can derive descendant keys. Having made a private key query for ID means that distinguishing ciphertexts encrypted to any ID^* that is a descendant of ID is trivial; but if the query at ID was restricted to depth d, distinguishing is only trivial if the depth of ID^* is at most d. All this discussion motivates the following definition for the restricted-delegation (IND-ID-RD-CPA) game played between an adversary $\mathcal{A}$ and a challenger, and stated for an ℓ-level HIBE.

Setup: The challenger runs the `Setup` algorithm and gives $\mathcal{A}$ the public parameters *params*, keeping *master-key* private.

Phase 1: Algorithm $\mathcal{A}$ adaptively issues queries of its choice, of the following types:

- Private key query for ID. The queried identity must have depth k with $1 \leq k \leq \ell$. The challenger runs $\texttt{KeyGen}(\mathsf{ID})$ using *master-key* and returns the resulting key d_{ID} to $\mathcal{A}$.
- Restricted private key query for ID, with restriction depth d. The queried identity must have depth k with $1 \leq k \leq \ell$. The depth restriction d must satisfy $k \leq d \leq \ell$. The challenger runs $\texttt{KeyGen}(\mathsf{ID})$ using *master-key* to obtain the private key d_{ID}, which it then restricts to depth d by running $\texttt{Restrict}(d_{\mathsf{ID}}, d)$; the result of this is returned to $\mathcal{A}$.

Challenge: Algorithm $\mathcal{A}$ chooses a challenge identity ID^* and two equal-length messages M_0 and M_1. Let the depth of ID^* be d^*. We require that $\mathcal{A}$ must not have made in Phase 1 a private key query at any ancestor of ID^*, or a restricted private key query for any pair (ID, d) where ID is an ancestor of ID^* and $d \geq d^*$. The challenger chooses $b \xleftarrow{\mathrm{R}} \{0,1\}$, computes $\mathsf{CT}^* \leftarrow \texttt{Encrypt}(\mathit{params}, \mathsf{ID}^*, M_b)$, and sends CT^* to $\mathcal{A}$.

Phase 2: Algorithm $\mathcal{A}$ issues additional queries as in Phase 1, with the following limitations:

- for a private key query, the queried identity ID must not be an ancestor of ID^*; and
- for a restricted private key query for a pair (ID, d), if ID is an ancestor of ID^* then d must be less than d^*.

The queries are handled by the challenger as in Phase 1.

Guess: Finally, $\mathcal{A}$ outputs its guess b' for b.

We define the advantage of the adversary in attacking a scheme $\mathcal{E}$ in the standard way:

$$\mathrm{Adv}_{\mathcal{E},\mathcal{A}} = \left|\Pr[b = b'] - 1/2\right| \ ,$$

with the probability taken over the coins of the adversary and challenger.

We make two observations about the definition above:

- When the hierarchy is limited to some depth ℓ (as it is for BBG) a key with delegation restricted to depth ℓ is the same as an unrestricted key, so the private key queries in game above are, strictly speaking, extraneous.
- When the adversary makes no restricted private key queries the IND-ID-RD-CPA game is identical to the IND-ID-CPA game. Thus a scheme that is secure with restricted delegation is also secure in the standard sense.

4.2 The Restricted-Delegation Game for Selective-ID HIBE

It is possible to turn the definition above into a selective-ID definition simply by having the adversary declare ID^* at the beginning of the game, before it sees the public parameters. Having made this transformation, however, we will find that the game can be substantially simplified.

Suppose ID^* has depth $d^* \leq \ell$. Now for any identity ID the following holds true:

- If ID is not an ancestor of ID^*, then a private key query at ID does not fall under the triviality restriction. Thus there is no sense in handling restricted private key queries for ID. If $\mathcal{A}$ is interested in a version of d_{ID} with restricted delegation, it can generate this on its own using `Restrict`.
- If ID *is* an ancestor of ID^*, then a (full) private key query at ID would violate the triviality restriction. Moreover, a restricted private key query at (ID, d) would not violate the restriction exactly when $d < d^*$. But now there is no sense in handling queries for depth other than $d^* - 1$; if the adversry wishes to restrict delegation further, it can again perform the restriction on its own using `Restrict`.

These observations, applied to the original game, yield the following simpler selective-ID restricted-delegation (IND-sID-RD-CPA) game, again played between an adversary $\mathcal{A}$ and a challenger and stated for an ℓ-level HIBE.

ID Selection: Adversary $\mathcal{A}$ chooses an identity ID^* to attack, and sends it to the challenger. Let the depth of ID^* be d^*, with $1 \leq d^* \leq k$.

Setup: The challenger runs the `Setup` algorithm and gives $\mathcal{A}$ the public parameters *params*, keeping *master-key* private.

Phase 1: Algorithm $\mathcal{A}$ adaptively issues queries, to which the challenger responds. In each query, $\mathcal{A}$ provides an identity ID for which it would like to receive a private key. The challenger runs $\texttt{KeyGen}(\mathsf{ID})$ using *master-key* and returns the resulting key d_{ID} to $\mathcal{A}$. If ID is not an ancestor of ID^*, the challenger provides d_{ID} to $\mathcal{A}$ unchanged. Otherwise, the challenger restricts d_{ID} to depth $d^* - 1$ by running $\texttt{Restrict}(d_{\mathsf{ID}}, d^* - 1)$ and provides the result of the restriction to $\mathcal{A}$. Algorithm $\mathcal{A}$ may not make a query at ID^*.

Challenge: Algorithm $\mathcal{A}$ sends to the challenger two equal-length messages M_0 and M_1. The challenger chooses $b \xleftarrow{R} \{0, 1\}$, computes $\mathsf{CT}^* \leftarrow \texttt{Encrypt}(\mathit{params}, \mathsf{ID}^*, M_b)$, and sends CT^* to $\mathcal{A}$.

Phase 2: Algorithm $\mathcal{A}$ issues additional queries and the challenger responds as in Phase 1.

Guess: Finally, $\mathcal{A}$ outputs its guess b' for b.

The adversary's advantage is defined as above. Once again, note that when $\mathcal{A}$ never makes a query for ID that is an ancestor of ID^* the game is the same as the standard IND-sID-CPA game.

5 Restricted Delegation in BBG

We now show that the BBG scheme has restricted delegation in the selective-ID setting, assuming BDHE is hard. For the queries allowed by the IND-sID-CPA game, we follow exactly the proof given

in [2]. Our addition to the proof handles queries for ID that is an ancestor of ID^*; we will note this addition as well.

One potential source of confusion: as typically stated, the ℓ-BDHI assumption includes the exponents g^{α^i} up to $i = \ell$, whereas the ℓ-BDHE assumption includes the exponents only up to $i = \ell - 1$ (and then again from $\ell + 1$ through 2ℓ. Our reduction thus uses the $(\ell + 1)$-BDHE assumption for an ℓ-level HIBE, which keeps the "hole" at index $\ell + 1$.

Theorem 5.1. *If $(\ell+1)$-BDHE holds for a group G as in Section 2 then BBG is secure for ℓ levels in an* IND-sID-RD-CPA *game on G.*

Proof. Given an adversary with advantage ϵ in the IND-sID-RD-CPA game with ℓ-level hierarchy, we build an algorithm $\mathcal{B}$ with advantage ϵ in distinguishing $(\ell + 1)$-BDHE. For a generator $g \in G$ and $\alpha \in \mathbb{Z}_p^*$, define $y_i = g^{(\alpha^i)}$. Algorithm $\mathcal{B}$ is given generators g and h along with values $y_1, \ldots, y_\ell$, $y_{\ell+2}, \ldots, y_{2\ell+2}$ as well as T, which is either the BDHE solution $e(y_{\ell+1}, h) = e(g, h)^{(\alpha^{\ell+1})}$ or random in G_1^*. Algorithm $\mathcal{B}$'s goal is to output 1 if T is the BDHE solution, 0 if T is random. It interacts with $\mathcal{A}$ as follows.

Initialization. Algorithm $\mathcal{A}$ outputs a challenge identity ID^*, which we treat as a depth-ℓ identity $(I_1^*, \ldots, I_\ell^*)$, padding with zero-entries as necessary. We also record the actual depth $d^* \le \ell$ of ID^*, i.e., the index of the last nonzero entry in $(I_1^*, \ldots, I_\ell^*)$.

Setup. Simulator $\mathcal{B}$ now picks a number of random exponents:

$$\gamma, \gamma_1, \ldots, \gamma_\ell, \delta \xleftarrow{\mathrm{R}} \mathbb{Z}_p \ ;$$

it uses these to generate the system parameters. First, it sets $g_2 \leftarrow y_\ell \cdot g^\gamma = g^{\gamma+\alpha^\ell}$. Next, it computes the hash generators: for each i, $1 \le i \le \ell$, it sets $h_i \leftarrow g^{\gamma_i}/y_{\ell-i+1}$, and, finally $g_3 \leftarrow g^\delta \cdot \prod_{i=1}^{\ell} y_{\ell-i+1}^{I_i^*}$. The parameters $(g, g_1, g_2, g_3, h_1, \ldots, h_\ell)$ arge given to $\mathcal{A}$; it is easy to see that they are properly distributed. The master secret $g_2^\alpha = (y_1)^\gamma (y_{\ell+1})$ is unknown to $\mathcal{B}$ because $y_{\ell+1}$ is.

Phase 1. Suppose $\mathcal{A}$ issues a private key query for $\mathsf{ID} = (I_1, \ldots, I_u) \in (\mathbb{Z}_p^*)^u$ with $u \le \ell$, and where ID is not a prefix of ID^*. Let $k \le u$ be the first index at which ID and ID^* differ. (If none such exists, ID is a prefix of ID^*.) Algorithm $\mathcal{B}$ constructs a secret key $d_{\mathsf{ID}|k}$, then uses `KeyGen` to derive d_{ID}, which it returns to $\mathcal{A}$.

To generate $d_{\mathsf{ID}|k}$, algorithm $\mathcal{B}$ chooses $\tilde{r} \xleftarrow{\mathrm{R}} \mathbb{Z}_p$. Define $r = \tilde{r} + \alpha^k/(I_k - I_k^*)$; while $\mathcal{B}$ cannot compute this, it *can* compute the private key with r as randomness, and a random choice of $\tilde{r}$ induces a random choice of r, so the resulting key will be properly distributed. To dispose first of the terms $(b_{k+1}, \ldots, b_\ell)$, we observe that, for $k + 1 \le i \le \ell$,

$$\begin{aligned} b_i = h_i^r = \left(g^{\gamma_i}/y_{\ell-i+1}\right)^r &= (g^{\gamma_i})^{\tilde{r}+\alpha^k/(I_k-I_k^*)} \cdot (y_{\ell-i+1})^{\tilde{r}+\alpha^k/(I_k-I_k^*)} \\ &= (g)^{\gamma_i\tilde{r}} (y_k)^{\gamma_i/(I_k-I_k^*)} \cdot (y_{\ell-i+1})^{\tilde{r}} (y_{\ell+(k-i)+1})^{1/(I_k-I_k^*)} \ , \end{aligned}$$

where in the last expression $\mathcal{B}$ knows each of the parenthesized terms — the term $y_{\ell+(k-i)+1}$ because the restriction $k+1 \le i \le \ell$ implies $\ell + (k-i) + 1 \in \{k+1, \ldots, \ell\}$. Similarly, $\mathcal{B}$ can compute a_1, since we have

$$a_1 = g^r = (g)^{\tilde{r}} \cdot (y_k)^{1/(I_K-I_k^*)} \ .$$

It thus remains only to show how $\mathcal{B}$ can compute a_0. But now observe:

$$\begin{aligned}
\left(h_1^{I_1} h_2^{I_2} \cdots h_k^{I_k} \cdot g_3\right)^r &= \Big(\prod_{i=1}^{k} (g^{\gamma_i}/y_{\ell-i+1})^{I_i} \times g^{\delta} \prod_{i=1}^{\ell} y_{\ell-i+1}^{I_i^*}\Big)^r \\
&= \Big(g^{\delta+\sum_{i=1}^k \gamma_i I_i}\Big)^r \Big(\prod_{i=1}^{k} y_{\ell-i+1}^{I_i^*-I_i} \times \prod_{i=k+1}^{\ell} y_{\ell-i+1}^{I_i^*}\Big)^r \\
&= \Big(g^{\delta+\sum_{i=1}^k \gamma_i I_i}\Big)^r \Big(y_{\ell-k+1}^{I_k^*-I_k} \cdot \prod_{i=k+1}^{\ell} y_{\ell-i+1}^{I_i^*}\Big)^r \\
&= \Big(g^{\delta+\sum_{i=1}^k \gamma_i I_i}\Big)^r \Big(y_{\ell-k+1}^{I_k^*-I_k} \cdot \prod_{i=k+1}^{\ell} y_{\ell-i+1}^{I_i^*}\Big)^r
\end{aligned}$$

where the terms with $i < k$ in the product equal 1 since for such i we have $I_i^* - I_i = 0$. Letting $t = \delta + \sum_{i=1}^k \gamma_i I_i$, we continue:

$$\begin{aligned}
\left(h_1^{I_1} h_2^{I_2} \cdots h_k^{I_k} \cdot g_3\right)^r &= \cdots \\
&= (g)^{t\tilde{r}} (y_k)^{t/(I_k - I_k^*)} (y_{\ell-k+1}^{I_k^*-I_k})^r \Big(\prod_{i=k+1}^{\ell} y_{\ell-i+1}^{I_i^*}\Big)^r \\
&= (g)^{t\tilde{r}} (y_k)^{t/(I_k - I_k^*)} (y_{\ell-k+1})^{(\tilde{r})(I_k^*-I_k)} (y_{\ell+1})^{-1} \\
&\quad \times \prod_{i=k+1}^{\ell} (y_{\ell-i+1})^{(\tilde{r})(I_i^*)} \times \prod_{i=k+1}^{\ell} (y_{\ell+(k-i)+1})^{I_i^*/(I_k^*-I_k)} \ .
\end{aligned}$$

All the terms of this last expression, except $(y_{\ell+1})^{-1}$, can be computed by $\mathcal{B}$; the terms $y_{\ell+(k-i)+1}$, in particular, are known to $\mathcal{B}$ by the range argument above. Let Z' be the known quantities, so that $\left(h_1^{I_1} h_2^{I_2} \cdots h_k^{I_k} \cdot g_3\right)^r = Z' \cdot (y_{\ell+1})^{-1}$.[1] But now we have

$$a_0 = g_2^{\alpha} \cdot \left(h_1^{I_1} h_2^{I_2} \cdots h_k^{I_k} \cdot g_3\right)^r = (y_1)^{\gamma} (y_{\ell+1}) \cdot Z' \cdot (y_{\ell+1})^{-1} = (y_1)^{\gamma} \cdot Z' \ ,$$

which $\mathcal{B}$ can compute. Thus $\mathcal{B}$ can answer $\mathcal{A}$'s query.

In the new case we handle, ID is a prefix of ID^* of length k such that $0 < k < d^*$. We show how $\mathcal{B}$ computes a private key for ID that is restricted up to depth $d^* - 1$. As before, it chooses $\tilde{r} \xleftarrow{R} \mathbb{Z}_p$, but this time it defines $r = \tilde{r} - \alpha^{d^*}/I_{d^*}^*$. We now show that $\mathcal{B}$ can compute (a_0, a_1) along with $(b_{k+1}, \ldots, b_{d^*-1})$ — that is, enough to delegate up to depth $d^* - 1$. Specifically, for $k + 1 \le i \le d^* - 1$, we have

$$\begin{aligned}
b_i = h_i^r &= \left(g^{\gamma_i}/y_{\ell-i+1}\right)^r = (g^{\gamma_i})^{\tilde{r} - \alpha^{d^*}/(I_k^*)} \cdot (y_{\ell-i+1})^{\tilde{r} + \alpha^{d^*}/(I_k - I_k^*)} \\
&= (g)^{\gamma_i \tilde{r}} (y_{d^*})^{-\gamma_i / I_k^*} \cdot (y_{\ell-i+1})^{\tilde{r}} (y_{\ell+(d^*-i)+1})^{-1/I_k^*} \ .
\end{aligned}$$

Here $\mathcal{B}$ can compute each term. In particular that $y_{\ell+(d^*-i)+1}$ is known to $\mathcal{B}$ because we have $\ell + (d^* - i) + 1 \in \{\ell + 2, \ldots, 2\ell\}$ when $i \le d^* - 1$. (When $i = d^*$, however, $y_{\ell+(d^*-i)+1}$ is $y_{\ell+1}$,

[1] We include more in Z' than Boneh, Boyen, and Goh include in their term Z, whence the change of notation.

so $\mathcal{B}$ couldn't compute b_{d^*}. This is as it should be, since otherwise $\mathcal{B}$ could compute the key for ID^*.) Next, $\mathcal{B}$ can compute a_1, since we have

$$a_1 = g^r = (g)^{\tilde{r}} \cdot (y_{d^*})^{-1/I_k^*} \ .$$

Finally, we consider a_0. For this we have

$$\begin{aligned}\left(h_1^{I_1} h_2^{I_2} \cdots h_k^{I_k} \cdot g_3\right)^r &= \left(g^{\delta + \sum_{i=1}^k \gamma_i I_i}\right)^r \left(\prod_{i=1}^{k} y_{\ell-i+1}^{I_i^* - I_i} \times \prod_{i=k+1}^{\ell} y_{\ell-i+1}^{I_i^*}\right)^r \\ &= (g^t)^r \Big(\prod_{i=k+1}^{\ell} y_{\ell-i+1}^{I_i^*} \Big)^r\end{aligned}$$

since $I_i = I_i^*$ for $1 \le i \le k$, and where we let $t = \delta + \sum_{i=1}^k \gamma_i I_i^*$. Continuing, we have

$$\begin{aligned}\left(h_1^{I_1} h_2^{I_2} \cdots h_k^{I_k} \cdot g_3\right)^r &= \cdots \\ &= (g)^{t\tilde{r}} (y_{d^*})^{-t/I_{d^*}} \times \prod_{i=k+1}^{\ell} y_{\ell-i+1}^{(\tilde{r})(I_i^*)} \times \prod_{i=k+1}^{\ell} y_{\ell+d^*-i+1}^{-I_i^*/I_{d^*}^*} \ .\end{aligned}$$

Now it is clear that $\mathcal{B}$ can calculate all the multiplicands except the last product, since the y-indexes are all in the range $\{1, \ldots, \ell\}$. As for the last product, noting that $k+1 \le d^* \le \ell$, we divide it into three cases: (1) $k+1 \le i \le d^*-1$; (2) $i = d^*$; and (3) $d^*+1 \le i \le \ell$. In the first case, we have $\ell + d^* - i + 1 \in \{\ell+2, \ldots, 2\ell\}$, so the y indexes are known to $\mathcal{B}$ and it can compute the terms. In the third case, we have $\ell + d^* - i + 1 \in \{2, \ldots, \ell\}$, and again $\mathcal{B}$ can compute the terms. In the second case, however, we have $y_{\ell+d^*-i+1}^{-I_i^*/I_{d^*}^*} = y_{\ell+1}^{-1}$. This term cancels out the term $y_{\ell+1}$ of g_2^α that $\mathcal{B}$ cannot compute; thus, since $\mathcal{B}$ can also compute the other term, $(y_1)^\gamma$, of g_2^α, it can compute a_0. Note that it is only in this case that we make use of the BDHE entries y_i with $i \ge \ell + 2$.

Challenge. When the adversary provides messages M_0 and M_1, algorithm $\mathcal{B}$ chooses $b \xleftarrow{\mathrm{R}} \{0,1\}$ and computes and responds with ciphertext

$$\mathsf{CT} \leftarrow \left(M_b \cdot T \cdot e(y_1, h^\gamma),\ h,\ h^{\delta + \sum_{i=1}^{\ell} \gamma_i I_i^*}\right) \ .$$

We argue that this is a valid encryption of M_b when T is the answer to the BDHE challenge, and the valid encryption of a random message when T is random. First, note that the randomness in the second component is properly distributed since h is unformly distributed in the BDHE challenge and nothing outside of CT seen by $\mathcal{A}$ depends on h. Next, the third component is properly formed, since

$$h_1^{I_1^*} h_2^{I_2^*} \cdots h_k^{I_k^*} \cdot g_3 = \prod_{i=1}^{\ell} (g^{\gamma_i} / y_{\ell-i+1})^{I_i^*} \times g^\delta \prod_{i=1}^{\ell} y_{\ell-i+1}^{I_i^*} = g^{\delta + \sum_{i=1}^{\ell} (\gamma_i)(I_i^*)} \ ,$$

and, letting $h = g^c$ for some unknown c, we have

$$\left(h_1^{I_1^*} h_2^{I_2^*} \cdots h_k^{I_k^*} \cdot g_3\right)^c = \left(g^{\delta + \sum_{i=1}^{\ell} (\gamma_i)(I_i^*)}\right)^c = h^{\delta + \sum_{i=1}^{\ell} (\gamma_i)(I_i^*)} \ .$$

Finally, for the same c as above, the correct message blinding factor would be

$$e(g_1, g_2)^c = e(y_1, y_\ell \cdot g^\gamma)^c = e(g, g)^{c\alpha^{\ell+1}} e(y_1, g)^{c\gamma} = e(g, h)^{\alpha^{\ell+1}} e(y_1, h)^\gamma \ .$$

Comparing this expression to the message blinding factor actually used in computing CT above, we see that if T equals $e(g, h)^{\alpha^{\ell+1}}$ the CT is a valid encryption of M_b; but if T is random then it is the encryption of a random message, and thus independent of b.

Phase 2. In Phase 2, algorithm $\mathcal{B}$ responds to $\mathcal{A}$'s queries as it did in Phase 1, above.

Guess. When $\mathcal{A}$ halts, outputting its guess b', algorithm $\mathcal{B}$ outputs 1, meaning that T is the answer to the BDHE challenge (i.e., $T = e(g, h)^{\alpha^{\ell+1}}$) if b equals b', and answers 0, meaning that T is random, otherwise.

When T is the BDHE answer, algorithm $\mathcal{A}$'s environment is perfectly simulated, so we have $\left|\Pr[b = b'] - 1/2\right| \geq \epsilon$. When T is random, $\mathcal{A}$ can do no better than guess, so we have $\left|\Pr[b = b'] - 1/2\right| = 0$. Thus

$$\left|\Pr[\mathcal{B} = 1 \mid T = e(g, h)^{\alpha^{\ell+1}}] - \Pr[\mathcal{B} = 1 \mid T \text{ is random}]\right| \geq |(1/2 \pm \epsilon) - 1/2| = \epsilon,$$

which completes the proof. □

Restricted Master Secret. Note that our proof handles the case where ID is a parent of ID^* even when the depth of ID is zero, i.e., ID is the empty or root identity. In this case, what is returned is a restricted master secret $(a_0, a_1, b_1, \ldots, b_{d^*-1})$ that can be used to compute any key of depth at most $d^* - 1$. Its first two components are of the form $(a_0, a_1) = (g_2^\alpha g_3^r, g^r)$.

References

[1] D. Boneh and X. Boyen. Efficient selective-ID secure identity based encryption without random oracles. In C. Cachin and J. Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 223–38. Springer-Verlag, May 2004.

[2] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In R. Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *LNCS*, pages 440–56. Springer-Verlag, May 2005. Full version: `http://ai.stanford.edu/~xb/eurocrypt05a/`.

[3] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In V. Shoup, editor, *Proceedings of Crypto 2005*, volume 3621 of *LNCS*, pages 258–275. Springer-Verlag, Aug. 2005.

[4] S. Galbraith. Pairings. In I. F. Blake, G. Seroussi, and N. Smart, editors, *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes*, chapter IX, pages 183–213. Cambridge University Press, 2005.

[5] K. Paterson. Cryptography from pairings. In I. F. Blake, G. Seroussi, and N. Smart, editors, *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes*, chapter X, pages 215–51. Cambridge University Press, 2005.