An extended abstract of this paper appears in R. Cramer, editor, Advances in Cryptology—EUROCRYPT 2005, volume 3493 of Lecture Notes in Computer Science, pages 440–456, Springer, 2005. Online as IACR ePrint Report 2005/015 or `http://www.cs.stanford.edu/~xb/eurocrypt05a/`.

Hierarchical Identity Based Encryption with Constant Size Ciphertext

Dan Boneh* dabo@cs.stanford.edu
Xavier Boyen† xb@boyen.org
Eu-Jin Goh* eujin@cs.stanford.edu

June 20, 2005

Abstract

We present a Hierarchical Identity Based Encryption (HIBE) system where the ciphertext consists of just three group elements and decryption requires only two bilinear map computations, *regardless of the hierarchy depth*. Encryption is as efficient as in other HIBE systems. We prove that the scheme is selective-ID secure in the standard model and fully secure in the random oracle model. Our system has a number of applications: it gives very efficient forward secure public key and identity based cryptosystems (with short ciphertexts), it converts the NNL broadcast encryption system into an efficient public key broadcast system, and it provides an efficient mechanism for encrypting to the future. The system also supports limited delegation where users can be given restricted private keys that only allow delegation to bounded depth. The HIBE system can be modified to support sublinear size private keys at the cost of some ciphertext expansion.

1 Introduction

An Identity Based Encryption (IBE) system [Sha84, BF01] is a public key system where the public key can be an arbitrary string such as an email address. A central authority uses a master key to issue private keys to identities that request them. Hierarchical IBE (HIBE) [HL02, GS02] is a generalization of IBE that mirrors an organizational hierarchy. An identity at level k of the hierarchy tree can issue private keys to its descendant identities, but cannot decrypt messages intended for other identities (details are given in Section 2.1). The first construction for HIBE is due to Gentry and Silverberg [GS02] where security is based on the Bilinear Diffie-Hellman (BDH) assumption in the random oracle model. A subsequent construction due to Boneh and Boyen [BB04a] gives an efficient (selective-ID secure) HIBE based on BDH without random oracles. In both constructions, the length of ciphertexts and private keys, as well as the time needed for decryption and encryption, grows linearly in the depth ℓ of the hierarchy.

There are currently two principal applications for HIBE. The first, due to Canetti, Halevi, and Katz [CHK03], is forward secure encryption. Forward secure encryption enables users to periodically update their private keys so that a message encrypted at period n cannot be read using a private key from period $n' > n$. To provide for $T = 2^t$ time periods, the CHK construction uses a HIBE of depth t where identities are *binary* vectors of length at most t. At time n, the encryptor encrypts using the identity corresponding to the n-th node of this depth t binary tree. Consequently,

*Stanford University. Supported by NSF.
†Voltage Inc., Palo Alto.

using previous HIBE systems [GS02, BB04a], ciphertexts in this forward secure construction are of size $O(t)$; private keys are of size $O(t^2)$ but can be reduced to size $O(t)$ by using updateable public storage. The second application for HIBE, due to Dodis and Fazio [DF02], is using HIBE to convert the NNL broadcast encryption system [NNL01] into a *public-key* broadcast system. Unfortunately, the resulting public-key broadcast system is no better than simpler constructions because ciphertext length in previous HIBE constructions is linear in the depth of the hierarchy.

Our Contribution. We present a HIBE system where the ciphertext size as well as the decryption cost are *independent* of the hierarchy depth ℓ. Ciphertexts in our HIBE system are always just three group elements and decryption requires only two bilinear map computations. Private keys in our basic system contain ℓ group elements as in previous HIBE constructions.

Our system gives a forward secure encryption system with short ciphertexts consisting of only three group elements, for any number $T = 2^t$ of time periods. With our basic HIBE system, the private key size in this forward secure encryption system is $O(t^2)$. In Section 4 we describe a hybrid system that borrows some features from the Boneh-Boyen HIBE [BB04a] and results in a forward secure encryption scheme where private key size is reduced to $O(t^{3/2})$ and ciphertext size is $O(\sqrt{t})$. By using updateable public storage as in CHK [CHK03], private key size in these systems can be further reduced to size $O(t)$ and $O(\sqrt{t})$ respectively. In addition, instantiating the Dodis-Fazio [DF02] system with our HIBE system results in a *public-key* broadcast system that is as efficient as the NNL subset difference method.

It is worth noting that private keys in our system *shrink* as the identity depth increases; this shrinkage is the opposite behavior from previous HIBE systems where private keys become larger as we descend deeper down the hierarchy tree. This behavior leads to "limited delegation" where an identity at depth k can be given a restricted private key that only lets it issue private keys to descendants of limited depth (as opposed to any descendant).

Security of our system is based on the Bilinear Diffie-Hellman Inversion assumption previously used in [BB04a, DY05, MSK02]. We describe the assumption in Section 2.3. In Section 3 we describe our HIBE system and prove its security in the selective identity model without using random oracles. We then observe that a selective-ID secure HIBE results in a fully secure HIBE in the random oracle model. In Sections 4 and 5 we discuss several extensions and applications of the system. For example, in addition to the applications already mentioned, we show how private keys can be further compressed to sublinear size and also describe an efficient mechanism for encrypting to the future.

2 Preliminaries

We briefly review the definition of HIBE and bilinear groups, and describe the Bilinear Diffie-Hellman Inversion assumption in such groups.

2.1 Fully Secure HIBE Systems

Like an Identity Based Encryption (IBE) system, a Hierarchical Identity Based Encryption (HIBE) system consists of four algorithms [HL02, GS02, BB04a]: `Setup`, `KeyGen`, `Encrypt`, `Decrypt`. In HIBE, however, identities are vectors; a vector of dimension k represents an identity at depth k. The `Setup` algorithm generates system parameters, denoted by *params*, and a master key *master-key*. We refer to the *master-key* as the private key at depth 0 and note that an IBE system is a HIBE where all identities are at depth 1. Algorithm `KeyGen` takes as input an identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$

at depth k and the private key $d_{\mathsf{ID}|k-1}$ of the parent identity $\mathsf{ID}_{|k-1} = (\mathrm{I}_1, \ldots, \mathrm{I}_{k-1})$ at depth $k-1$, and then outputs the private key d_{ID} for identity ID. The encryption algorithm encrypts messages for an identity using *params* and the decryption algorithm decrypts ciphertexts using the private key.

Chosen ciphertext security for HIBE systems is defined under a chosen identity attack where the adversary is allowed to adaptively chose the public key on which it will be challenged. More precisely, HIBE security (IND-ID-CCA) is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

Setup: The challenger $\mathcal{C}$ runs the `Setup` algorithm and gives $\mathcal{A}$ the resulting system parameters *params*, keeping the *master-key* to itself.

Phase 1: $\mathcal{A}$ adaptively issues queries $q_1, \ldots, q_m$ where query q_i is one of the following:

– Private key query $\langle \mathsf{ID}_i \rangle$. $\mathcal{C}$ responds by running algorithm `KeyGen` to generate the private key d_i corresponding to the public key $\langle \mathsf{ID}_i \rangle$ and sends d_i to $\mathcal{A}$.

– Decryption query $\langle \mathsf{ID}_i, C_i \rangle$. $\mathcal{C}$ responds by running algorithm `KeyGen` to generate the private key d corresponding to ID_i. It then runs algorithm `Decrypt` to decrypt the ciphertext C_i using the private key d and sends the resulting plaintext to $\mathcal{A}$.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is over, it outputs an identity ID^* and two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ on which it wishes to be challenged. The only restriction is that $\mathcal{A}$ did not previously issue a private key query for ID^* or a prefix of ID^*. $\mathcal{C}$ picks a random bit $b \in \{0,1\}$ and sets the challenge ciphertext to $\mathsf{CT} = \mathtt{Encrypt}(params, \mathsf{ID}^*, M_b)$, which is sent to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ issues additional queries $q_{m+1}, \ldots, q_n$ where q_i is one of:

– Private key query $\langle \mathsf{ID}_i \rangle$ where $\mathsf{ID}_i \neq \mathsf{ID}^*$ and ID_i is not a prefix of ID^*.
– Decryption query $\langle C_i \rangle \neq \langle C \rangle$ for ID^* or any prefix of ID^*.

In both cases, $\mathcal{C}$ responds as in Phase 1. These queries may be adaptive.

Guess: Finally, the adversary outputs a guess $b' \in \{0,1\}$ and wins if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-ID-CCA adversary. We define the advantage of the adversary $\mathcal{A}$ in attacking the scheme $\mathcal{E}$ as

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

The probability is over the random bits used by the challenger and the adversary.

Canetti, Halevi, and Katz [CHK03, CHK04] define a weaker notion of security in which the adversary commits ahead of time to the public key it will attack. We refer to this notion as selective identity, chosen ciphertext secure HIBE (IND-sID-CCA). The game is exactly the same as IND-ID-CCA except that the adversary $\mathcal{A}$ discloses to the challenger the target identity ID^* before the **Setup** phase. The restrictions on private key queries from phase 2 also hold in phase 1.

Definition 2.1. We say that a HIBE system $\mathcal{E}$ is $(t, q_{\mathsf{ID}}, q_C, \epsilon)$-secure if for any t-time IND-ID-CCA (respectively IND-sID-CCA) adversary $\mathcal{A}$ that makes at most q_{ID} chosen private key queries and at most q_C chosen decryption queries, we have that $\mathrm{Adv}_{\mathcal{E},\mathcal{A}} < \epsilon$. As shorthand, we say that $\mathcal{E}$ is $(t, q_{\mathsf{ID}}, q_C, \epsilon)$-IND-ID-CCA (resp. IND-sID-CCA) secure.

Semantic Security. As usual, we define chosen plaintext security for a HIBE system as in the preceding game, except that the adversary is not allowed to issue any decryption queries. The adversary may still issue adaptive private key queries. This security notion is termed as IND-ID-CPA (or IND-sID-CPA in the case of a selective identity adversary).

Definition 2.2. We say that a HIBE system $\mathcal{E}$ is $(t, q_{\mathsf{ID}}, \epsilon)$-IND-ID-CPA secure (resp. IND-sID-CPA) if $\mathcal{E}$ is $(t, q_{\mathsf{ID}}, 0, \epsilon)$-IND-ID-CCA secure (resp. IND-sID-CCA).

2.2 Bilinear Groups

We briefly review bilinear maps and bilinear map groups. We use the following notation [Jou00, BLS01]:

1. $\mathbb{G}$ and $\mathbb{G}_1$ are two (multiplicative) cyclic groups of prime order p;
2. g is a generator of $\mathbb{G}$.
3. e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$.

Let $\mathbb{G}$ and $\mathbb{G}_1$ be two groups as above. A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ with the properties:

1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group action in $\mathbb{G}$ can be computed efficiently and there exists both a group $\mathbb{G}_1$ and an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ as above.

2.3 Bilinear Diffie-Hellman Inversion Assumption

Let $\mathbb{G}$ be a bilinear group of order p. Let $w \in \mathbb{G}$ be a generator and $\beta \in \mathbb{Z}_p^*$. The ℓ-th Bilinear Diffie-Hellman Inversion problem [BB04a, MSK02], denoted ℓ-BDHI, is as follows:

$$\ell\text{-BDHI} : \quad \text{given} \quad w, w^{\beta}, w^{(\beta^2)}, \ldots, w^{(\beta^\ell)} \quad \text{compute} \quad e(w, w)^{1/\beta} \tag{1}$$

Loosely speaking, the ℓ-BDHI assumption in $\mathbb{G}$ says that no efficient algorithm can solve ℓ-BDHI in $\mathbb{G}$ with non-negligible probability.

Weak BDHI Assumption. Our security theorems can be shown under a slightly weaker assumption, which we call Weak BDHI, and denote ℓ-wBDHI. This assumption is weaker in the sense that the ℓ-wBDHI assumption holds in any group where ℓ-BDHI holds, though the converse is not known to be true. To state the strongest results, we will use Weak BDHI throughout the paper.

Let g and h be two random generators of $\mathbb{G}$. Let α be a random number in $\mathbb{Z}_p^*$. We define the (equivalent) ℓ-wBDHI and ℓ-wBDHI* problems as follows:

$$\ell\text{-wBDHI} : \quad \text{given} \quad g, h, g^{\alpha}, g^{(\alpha^2)}, \ldots, g^{(\alpha^\ell)} \quad \text{compute} \quad e(g, h)^{1/\alpha} \tag{2}$$

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

These two problems are equivalent under a linear time reduction. The wBDHI problem (2) is seen to be more naturally related to the BDHI problem (1), but we will, for notational convenience, base our proofs on the wBDHI* problem (3).

It is easy to see that an algorithm for ℓ-wBDHI or ℓ-wBDHI* in $\mathbb{G}$ gives an algorithm for ℓ-BDHI with a tight reduction. Indeed, given a ℓ-BDHI problem instance $(w, w_1, \ldots, w_\ell)$, define the ℓ-wBDHI* instance $(w_\ell, h, w_{\ell-1}, \ldots, w_1, w)$ where $h = w_\ell^r$ for some random exponent $r \in \mathbb{Z}_p^*$. Let T' be the solution to this ℓ-wBDHI* problem instance, then $T = (T')^{1/r}$ is the solution to the original ℓ-BDHI instance.

We now define precisely the computational and decisional ℓ-wBDHI assumptions. For convenience, we define them in reference to the ℓ-wBDHI* problem.

As shorthand, let $y_i = g^{(\alpha^i)} \in \mathbb{G}^*$. An algorithm $\mathcal{A}$ has advantage ϵ in solving the ℓ-wBDHI* problem in $\mathbb{G}$ if

$$\Pr\left[\mathcal{A}(g, h, y_1, \ldots, y_\ell) = e(g,h)^{(\alpha^{\ell+1})}\right] \geq \epsilon,$$

where the probability is over the random choice of generators g, h in $\mathbb{G}^*$, the random choice of α in $\mathbb{Z}_p^*$, and the random bits used by $\mathcal{A}$. The decisional version of the ℓ-wBDHI* problem in $\mathbb{G}$ is defined in the usual manner, as follows. Let $\vec{y}_{g,\alpha,\ell} = (y_1, \ldots, y_\ell)$. An algorithm $\mathcal{B}$ that outputs $b \in \{0,1\}$ has advantage ϵ in solving Decision ℓ-wBDHI* in $\mathbb{G}$ if

$$\left| \Pr\left[\mathcal{B}(g, h, \vec{y}_{g,\alpha,\ell}, e(g,h)^{(\alpha^{\ell+1})}) = 0\right] - \Pr\left[\mathcal{B}(g, h, \vec{y}_{g,\alpha,\ell}, T) = 0\right] \right| \geq \epsilon,$$

where the probability is over the random choice of generators g, h in $\mathbb{G}^*$, the random choice of α in $\mathbb{Z}_p^*$, the random choice of $T \in \mathbb{G}_1^*$, and the random bits consumed by $\mathcal{B}$. We refer to the distribution on the left as $\mathcal{P}_{wBDHI^*}$ and the distribution on the right as $\mathcal{R}_{wBDHI^*}$.

Definition 2.3. We say that the (Decision) (t, ϵ, ℓ)-wBDHI assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the (Decision) ℓ-wBDHI* problem in $\mathbb{G}$.

For conciseness we occasionally drop the t and ϵ and simply refer to the (Decision) ℓ-wBDHI assumption in $\mathbb{G}$. As mentioned above, the computational and decisional ℓ-BDHI assumptions in $\mathbb{G}$ respectively imply the computational and decisional ℓ-wBDHI assumptions in $\mathbb{G}$. In Appendices A.2 and A.3, we show that a broad class of assumptions, including the ℓ-wBDHI assumption, hold in generic bilinear groups [Sho97].

3 A HIBE System with Constant Size Ciphertext

Let $\mathbb{G}$ be a bilinear group of prime order p and let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ be a bilinear map. For now, we assume that public keys (that is, identities ID) at depth k are vectors of elements in $(\mathbb{Z}_p^*)^k$. We write $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$. The j-th component corresponds to the identity at level j. We later extend the construction to public keys over $\{0,1\}^*$ by first hashing each component I_j using a collision resistant hash $H : \{0,1\}^* \to \mathbb{Z}_p^*$. We also assume that the messages to be encrypted are elements in $\mathbb{G}_1$. The HIBE system works as follows:

`Setup`(ℓ)**:** To generate system parameters for an HIBE of maximum depth ℓ, select a random generator $g \in \mathbb{G}$, a random $\alpha \in \mathbb{Z}_p$, and set $g_1 = g^\alpha$. Next, pick random elements $g_2, g_3, h_1, \ldots, h_\ell \in \mathbb{G}$. The public parameters and the master key are

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

`KeyGen`$(d_{\mathsf{ID}|k-1}, \mathsf{ID})$**:** To generate a private key d_{ID} for an identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$ of depth $k \leq \ell$, using the master secret, pick a random $r \in \mathbb{Z}_p$ and output

$$d_{\mathsf{ID}} = \left(g_2^\alpha \cdot (h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3)^r, \quad g^r, \quad h_{k+1}^r, \quad \ldots, \quad h_\ell^r\right) \in \mathbb{G}^{2+\ell-k}.$$

Note that d_{ID} becomes shorter as the depth of ID increases.

The private key for ID can be generated incrementally, given a private key for the parent identity $\mathsf{ID}_{|k-1} = (\mathrm{I}_1, \ldots, \mathrm{I}_{k-1}) \in (\mathbb{Z}_p^*)^{k-1}$, as required. Indeed, let

$$d_{\mathsf{ID}|k-1} = \left(g_2^{\alpha} \cdot \left(h_1^{\mathrm{I}_1} \cdots h_{k-1}^{\mathrm{I}_{k-1}} \cdot g_3\right)^{r'}, \quad g^{r'}, \quad h_k^{r'}, \ldots, h_\ell^{r'} \right) = (a_0, a_1, b_k, \ldots, b_\ell)$$

be the private key for $\mathsf{ID}_{|k-1}$. To generate d_{ID}, pick a random $t \in \mathbb{Z}_p$ and output

$$d_{\mathsf{ID}} = \left(a_0 \cdot b_k^{\mathrm{I}_k} \cdot \left(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3\right)^{t}, \quad a_1 \cdot g^t, \quad b_{k+1} \cdot h_{k+1}^t, \quad \ldots, \quad b_\ell \cdot h_\ell^t \right).$$

This private key is a properly distributed private key for $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$ for $r = r' + t \in \mathbb{Z}_p$.

`Encrypt`$(params, \mathsf{ID}, M)$**:** To encrypt a message $M \in \mathbb{G}_1$ under the public key $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$, pick a random $s \in \mathbb{Z}_p$ and output

$$\mathsf{CT} = \left(e(g_1, g_2)^s \cdot M, \quad g^s, \quad \left(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3\right)^s \right) \in \mathbb{G}_1 \times \mathbb{G}^2.$$

`Decrypt`$(d_{\mathsf{ID}}, \mathsf{CT})$**:** Consider an identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$. To decrypt a given ciphertext $\mathsf{CT} = (A, B, C)$ using the private key $d_{\mathsf{ID}} = (a_0, a_1, b_{k+1} \ldots, b_\ell)$, output

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

Indeed, for a valid ciphertext, we have

$$\frac{e(a_1, C)}{e(B, a_0)} = \frac{e\left(g^r, \ (h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3)^s\right)}{e\left(g^s, \ g_2^{\alpha}(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3)^r\right)} = \frac{1}{e(g, g_2)^{s\alpha}} = \frac{1}{e(g_1, g_2)^s}.$$

Observe that for identities at any depth, the ciphertext contains only 3 elements and decryption takes only 2 pairings. In previous HIBE systems, ciphertext size and decryption time grow linearly in the identity depth. Also, note that $e(g_1, g_2)$ used for encryption can be precomputed (or substituted for g_2 in the system parameters) so that encryption does not require any pairings.

3.1 Security

We first show that our HIBE scheme is selective identity secure (IND-sID-CPA) under the decisional Bilinear Diffie-Hellman Inversion assumption. As mentioned in Section 2.3, we use a slightly weaker assumption called the decisional Weak BDHI. We later describe how to provide both chosen ciphertext security (IND-sID-CCA) and full HIBE security (IND-ID-CCA).

Theorem 3.1. *Let $\mathbb{G}$ be a bilinear group of prime order p. Suppose the Decision (t, ϵ, ℓ)-wBDHI assumption holds in $\mathbb{G}$. Then the previously defined ℓ-HIBE system is (t', q_S, ϵ)-selective identity, chosen plaintext* (IND-sID-CPA) *secure for arbitrary ℓ, q_S, and $t' < t - \Theta(\tau\, \ell\, q_S)$, where τ is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the ℓ-HIBE system. Using $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the Decision ℓ-wBDHI* problem in $\mathbb{G}$.

For a generator $g \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p^*$ let $y_i = g^{(\alpha^i)} \in \mathbb{G}$. Algorithm $\mathcal{B}$ is given as input a random tuple $(g, h, y_1, \ldots, y_\ell, T)$ that is either sampled from $\mathcal{P}_{wBDHI^*}$ (where $T = e(g,h)^{(\alpha^{\ell+1})}$) or from $\mathcal{R}_{wBDHI^*}$ (where T is uniform and independent in $\mathbb{G}_1^*$). Algorithm $\mathcal{B}$'s goal is to output 1 when the input tuple is sampled from $\mathcal{P}_{wBDHI^*}$ and 0 otherwise. Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a selective identity game as follows:

Initialization. The selective identity game begins with $\mathcal{A}$ first outputting an identity $\mathsf{ID}^* = (\mathrm{I}_1^*, \ldots, \mathrm{I}_m^*) \in (\mathbb{Z}_p^*)^m$ of depth $m \le \ell$ that it intends to attack. If $m < \ell$ then $\mathcal{B}$ pads ID^* with $\ell - m$ zeroes on the right to make ID^* a vector of length ℓ. Hence, from here we assume that ID^* is a vector of length ℓ.

Setup. To generate the system parameters, algorithm $\mathcal{B}$ picks a random γ in $\mathbb{Z}_p$ and sets $g_1 = y_1 = g^\alpha$ and $g_2 = y_\ell \cdot g^\gamma = g^{\gamma + (\alpha^\ell)}$. Next, $\mathcal{B}$ picks random $\gamma_1, \ldots, \gamma_\ell$ in $\mathbb{Z}_p$ and sets $h_i = g^{\gamma_i}/y_{\ell-i+1}$ for $i = 1, \ldots, \ell$. Algorithm $\mathcal{B}$ also picks a random δ in $\mathbb{Z}_p$ and sets $g_3 = g^\delta \cdot \prod_{i=1}^{\ell} y_{\ell-i+1}^{I_i^*}$.

Finally, $\mathcal{B}$ gives $\mathcal{A}$ the system parameters $params = (g, g_1, g_2, g_3, h_1, \ldots, h_\ell)$. Observe that all these values are distributed uniformly and independently in $\mathbb{G}$ as required. The master key corresponding to these system parameters is $g_2^\alpha = g^{\alpha(\alpha^\ell + \gamma)} = y_{\ell+1} y_1^\gamma$, which is unknown to $\mathcal{B}$ since $\mathcal{B}$ does not have $y_{\ell+1}$.

Phase 1. $\mathcal{A}$ issues up to q_S private key queries. Consider a query for the private key corresponding to $\mathsf{ID} = (\mathrm{I}_1, \ldots, I_u) \in (\mathbb{Z}_p^*)^u$ where $u \le \ell$. The only restriction is that ID is not ID^* or a prefix of ID^*. This restriction ensures that there exists a $k \in \{1, \ldots, u\}$ such that $\mathrm{I}_k \ne \mathrm{I}_k^*$ (otherwise, ID would be a prefix of ID^*); we set k such that it is the smallest such index. To respond to the query, algorithm $\mathcal{B}$ first derives a private key for the identity $(\mathrm{I}_1, \ldots, \mathrm{I}_k)$ from which it then constructs a private key for the requested identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k, \ldots, \mathrm{I}_u)$.

To generate the private key for identity $(\mathrm{I}_1, \ldots, \mathrm{I}_k)$, $\mathcal{B}$ first picks a random $\tilde{r}$ in $\mathbb{Z}_p$. We pose $r = \frac{\alpha^k}{(\mathrm{I}_k - \mathrm{I}_k^*)} + \tilde{r} \in \mathbb{Z}_p$. Next, $\mathcal{B}$ generates the private key

$$\left(g_2^\alpha \cdot (h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} g_3)^r, \;\; g^r, \;\; h_{k+1}^r, \;\; \ldots, \;\; h_\ell^r \right), \tag{4}$$

which is a properly distributed private key for the identity $(\mathrm{I}_1, \ldots, \mathrm{I}_k)$. We show that $\mathcal{B}$ can compute all elements of this private key given the values at its disposal. We use the fact that $y_i^{(\alpha^j)} = y_{i+j}$ for any i, j.

To generate the first component of the private key, first observe that

$$(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} g_3)^r = \left(g^{\delta + \sum_{i=1}^k \mathrm{I}_i \gamma_i} \cdot \prod_{i=1}^{k-1} y_{\ell-i+1}^{(\mathrm{I}_i^* - \mathrm{I}_i)} \cdot y_{\ell-k+1}^{(\mathrm{I}_k^* - \mathrm{I}_k)} \cdot \prod_{i=k+1}^{\ell} y_{\ell-i+1}^{\mathrm{I}_i^*} \right)^r. \tag{5}$$

Let Z denote the product of the first, second, and fourth terms. That is,

$$Z = \left(g^{\delta + \sum_{i=1}^k \mathrm{I}_i \gamma_i} \cdot \underbrace{\prod_{i=1}^{k-1} y_{\ell-i+1}^{(\mathrm{I}_i^* - \mathrm{I}_i)}}_{=1} \cdot \prod_{i=k+1}^{\ell} y_{\ell-i+1}^{\mathrm{I}_i^*} \right)^r.$$

Note that the second term in Z equals 1 because $\mathrm{I}_i = \mathrm{I}_i^*$ for all $i < k$. One can verify that $\mathcal{B}$ can compute all the terms in Z given the values at its disposal. Next, observe that the third term in Eq (5), namely $y_{\ell-k+1}^{r(\mathrm{I}_k^* - \mathrm{I}_k)}$, is:

$$y_{\ell-k+1}^{r(\mathrm{I}_k^* - \mathrm{I}_k)} = y_{\ell-k+1}^{\tilde{r}(\mathrm{I}_k^* - \mathrm{I}_k)} \cdot y_{\ell-k+1}^{(\mathrm{I}_k^* - \mathrm{I}_k)\frac{\alpha^k}{(\mathrm{I}_k - \mathrm{I}_k^*)}} = y_{\ell-k+1}^{\tilde{r}(\mathrm{I}_k^* - \mathrm{I}_k)} / y_{\ell+1}.$$

Hence, the first component in the private key (4) is equal to:

$$g_2^{\alpha}(h_1^{\mathrm{I}_1}\cdots h_k^{\mathrm{I}_k} g_3)^r = (y_{\ell+1}y_1^{\gamma})\cdot Z\cdot (y_{\ell-k+1}^{\tilde{r}(\mathrm{I}_k^*-\mathrm{I}_k)}/y_{\ell+1}) = y_1^{\gamma}\cdot Z\cdot y_{\ell-k+1}^{\tilde{r}(\mathrm{I}_k^*-\mathrm{I}_k)}.$$

Since $y_{\ell+1}$ cancels out, all the terms in this expression are known to $\mathcal{B}$. Thus, $\mathcal{B}$ can compute the first private key component.

The second component, g^r, is $y_k^{1/(\mathrm{I}_k-\mathrm{I}_k^*)}g^{\tilde{r}}$ which $\mathcal{B}$ can compute. Similarly, the remaining elements $h_{k+1}^r,\ldots,h_\ell^r$ can be computed by $\mathcal{B}$ since they do not involve a $y_{\ell+1}$ term. Thus, $\mathcal{B}$ can derive a valid private key for $(\mathrm{I}_1,\ldots,\mathrm{I}_k)$. Algorithm $\mathcal{B}$ uses this private key to derive a private key for the descendant identity ID and gives $\mathcal{A}$ the result.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs two messages $M_0, M_1 \in \mathbb{G}_1$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ picks a random bit $b \in \{0,1\}$ and responds with the challenge ciphertext

$$\mathsf{CT} = (M_b\cdot T\cdot e(y_1, h^{\gamma}),\ \ h,\ \ h^{\delta+\sum_{i=1}^{\ell}\mathrm{I}_i^*\gamma_i})$$

where h and T are from the input tuple given to $\mathcal{B}$. First note that if $h = g^c$ (for some unknown c in $\mathbb{Z}_p$) then

$$h^{\delta+\sum_{i=1}^{\ell}\mathrm{I}_i^*\gamma_i} = \left(\prod_{i=1}^{\ell}(g^{\gamma_i}/y_{\ell-i+1})^{\mathrm{I}_i^*}\cdot(g^{\delta}\prod_{i=1}^{\ell}y_{\ell-i+1}^{\mathrm{I}_i^*})\right)^c = (h_1^{\mathrm{I}_1^*}\cdots h_\ell^{\mathrm{I}_\ell^*}g_3)^c, \quad \text{and}$$

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

Therefore, if $T = e(g,h)^{(\alpha^{\ell+1})}$ (i.e., when the input tuple is sampled from $\mathcal{P}_{wBDHI^*}$), then the challenge ciphertext is a valid encryption of M_b under the original (unpadded) identity $\mathsf{ID}^* = (\mathrm{I}_1^*,\ldots,\mathrm{I}_m^*)$ chosen by the adversary, since

$$\begin{aligned}\mathsf{CT} &= \big(\ M_b\cdot e(g_1,g_2)^c,\quad g^c,\quad (h_1^{\mathrm{I}_1^*}\cdots h_m^{\mathrm{I}_m^*}\cdots h_\ell^{\mathrm{I}_\ell^*}g_3)^c\ \big)\\ &= \big(\ M_b\cdot e(g_1,g_2)^c,\quad g^c,\quad (h_1^{\mathrm{I}_1^*}\cdots h_m^{\mathrm{I}_m^*}g_3)^c\ \big).\end{aligned}$$

On the other hand, when T is uniform and independent in $\mathbb{G}_1^*$ (when the input tuple is sampled from $\mathcal{R}_{wBDHI^*}$), CT is independent of b in the adversary's view.

Phase 2. $\mathcal{A}$ issues queries not issued in Phase 1. $\mathcal{B}$ responds as before.

Guess. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. Algorithm $\mathcal{B}$ concludes its own game by outputting a guess as follows. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning $T = e(g,h)^{(\alpha^{\ell+1})}$. Otherwise, it outputs 0 meaning T is random in $\mathbb{G}_1$.

When the input tuple is sampled from $\mathcal{P}_{wBDHI^*}$ (where $T = e(g,h)^{(\alpha^{\ell+1})}$), then $\mathcal{A}$'s view is identical to its view in a real attack game and therefore $\mathcal{A}$ satisfies $|\Pr[b=b'] - 1/2| \geq \epsilon$. When the input tuple is sampled from $\mathcal{R}_{wBDHI^*}$ (where T is uniform in $\mathbb{G}_1^*$) then $\Pr[b=b'] = 1/2$. Therefore, with g,h uniform in $\mathbb{G}$, α uniform in $\mathbb{Z}_p$, and T uniform in $\mathbb{G}_1$ we have that

$$\left|\Pr\left[\mathcal{B}\big(g,h,\vec{y}_{g,\alpha,\ell},e(g,h)^{(\alpha^{\ell+1})}\big)=0\right] \quad - \quad \Pr\left[\mathcal{B}\big(g,h,\vec{y}_{g,\alpha,\ell},T\big)=0\right]\right| \quad \geq \quad |(1/2\pm\epsilon)-1/2| \quad = \epsilon$$

as required, which completes the proof of the theorem. $\square$

Chosen Ciphertext Security. Canetti et al. [CHK04] show a general method of building an IND-sID-CCA secure ℓ-HIBE from a IND-sID-CPA secure $\ell+1$-HIBE. A more efficient construction is given by Boneh and Katz [BK05]. Applying either method to our HIBE construction results in a IND-sID-CCA secure ℓ-HIBE for arbitrary ℓ where the ciphertext length is independent of the hierarchy height.

Arbitrary Identities. We can extend our HIBE to handle arbitrary identities $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_\ell)$ with $\mathrm{I}_i \in \{0,1\}^*$ for $i = 1, \ldots, \ell$ by hashing each I_i with a collision resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$ during key generation and encryption. A standard argument shows that if the original HIBE scheme is IND-sID-CCA secure, then so is the HIBE scheme using H.

3.2 Full HIBE Security

Theorem 3.1 shows that our HIBE system is selective-ID secure without random oracles. Thus, the system is secure when the adversary commits ahead of time to the identity he intends to attack. Boneh and Boyen [BB04a] observed that IBE systems that are selective-ID secure are also fully secure (i.e., secure against adversaries that adaptively select the identity to attack) as long as one hashes the identity prior to using it. The reduction, however, is not tight. Let $H : \{0,1\}^* \to \{0,1\}^d$ be a hash function (where, e.g., $d = 160$ bits). Assuming H is collision resistant, the reduction introduces a 2^d multiplicative security loss factor in the standard model. When H is viewed as a random oracle, the reduction introduces a q_H multiplicative security loss factor where q_H is the number of the hash oracle queries issued by the adversary.

A similar observation applies to HIBE systems. Let $\mathcal{E}$ be a selective-ID secure HIBE of depth ℓ. Let $\mathcal{E}_H$ be an HIBE system where an identity $\mathsf{ID} = (I_1, \ldots, I_k)$ is hashed to $\mathsf{ID}_H = (H(I_1), \ldots, H(I_k))$ before using it in `KeyGen` and `Encrypt`. Then, if H is collision resistant, it follows that $\mathcal{E}_H$ is a fully secure HIBE, but the reduction introduces a loss factor of $2^{\ell d}$. In the random oracle model, $\mathcal{E}_H$ is a fully secure HIBE and the reduction introduces a loss factor of q_H^ℓ.

We remark that in the random oracle model, the public parameters are of constant size and contain only the two group elements (g, g_1); the other parameters $(g_2, g_3, h_1, \ldots, h_\ell)$ need not be specified as they can be derived by applying the oracle on a predetermined input string.

We also note that the construction of Waters [Wat05], for a fixed depth ℓ, applied to our HIBE could give a fully secure constant ciphertext HIBE with a polynomial time reduction to the underlying complexity assumption in the standard model. The resulting private keys are much larger, namely of size $d\ell$, as opposed to ℓ in our system.

4 Extensions

We discuss a number of extensions to the HIBE system of the previous section.

4.1 Limited Delegation

Let $d_{\mathsf{ID}} = (a_0, a_1, b_k, \ldots, b_\ell)$ be the private key for the identity ID. Note that the `Decrypt` algorithm uses only the terms a_0 and a_1, and the `KeyGen` algorithm uses only the remaining terms $b_k, \ldots, b_\ell$.

By removing any number of $b_k, \ldots, b_\ell$, an identity ID at depth k can be given a restricted private key that only lets it issue private keys to descendants of bounded depth. For example, if the private key for ID only contains b_k, b_{k+1}, b_{k+2} (instead of all $b_k, \ldots, b_\ell$), then ID can only issue private keys for three generations of descendants, and those descendants' private keys will be limited even further.

4.2 HIBE with Short Private Keys

Certain applications, such as the time lock encryption (to be described in Section 5), are better served by using a HIBE system with short private keys rather than ciphertexts. We show how to construct a HIBE system whose private key size grows only sublinearly with hierarchy depth.

The idea is to construct a hybrid of the HIBE in Section 3 and the Boneh-Boyen HIBE [BB04a]. Recall that in the former system the private key shrinks as the identity depth increases, while in the latter system the private key grows with the depth of an identity. The hybrid is based on the algebraic similarities between both systems, and exploits their opposite behavior with regard to private key size, to ensure that no private key ever contains more than $O(\sqrt{\ell})$ group elements.

Specifically, for $\omega \in [0,1]$, the hybrid scheme achieves $O(\ell^{\omega} + \ell^{1-\omega})$ private key size and $O(\ell^{\omega})$ ciphertext size at every level in a hierarchy of depth ℓ. The setting $\omega = 0$ corresponds to our HIBE, whereas $\omega = 1$ corresponds to the Boneh-Boyen HIBE [BB04a]. The most efficient hybrids are obtained when $\omega \in [0, 1/2]$. For example, when $\omega = 1/2$, private keys and ciphertexts are of size $O(\sqrt{\ell})$.

Hybrid Scheme. As before, we assume a bilinear group $\mathbb{G}$ and a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$, where $\mathbb{G}$ and $\mathbb{G}_1$ have prime order p. Let $\ell_1 = \lceil \ell^{\omega} \rceil$ and $\ell_2 = \lceil \ell^{1-\omega} \rceil$. The basic idea is to partition levels of the hierarchy into ℓ_1 consecutive groups of size ℓ_2. Within each group we use the system of Section 3. Between groups we use the Boneh-Boyen HIBE [BB04a].

Let $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$ be an identity of depth $k \le \ell$. We will represent ID as a pair (k, I) where $\mathrm{I} \in (\mathbb{Z}_p^*)^{\ell_1 \times \ell_2}$ is an $\ell_1 \times \ell_2$ matrix filled using the elements $\mathrm{I}_1, \ldots, \mathrm{I}_k$ in typographic order: one row at a time starting from the top, in each row starting from the left (note that $\ell_1 \cdot \ell_2 \ge \ell \ge k$; the unfilled matrix entries are undefined). For convenience, we decompose the indices $k = 1, \ldots, \ell$ into row-column pairs (k_1, k_2) such that $k = \ell_2 \cdot (k_1 - 1) + k_2$ where $k_1, k_2 > 0$. For shorthand, we write $(k_1, k_2) = k$. It follows that in the above matrix representation of ID we have $\mathrm{I}_{(i_1,i_2)} = \mathrm{I}_i$ for all $i = 1, \ldots, k$. Or, pictorially, for an ID at the maximum depth ℓ with $\mathrm{I} = \mathrm{I}_1, \ldots, \mathrm{I}_\ell$ and $\ell = \ell_1 \ell_2$:

$$\mathrm{I} \;=\; \begin{pmatrix} \mathrm{I}_1 & \mathrm{I}_2 & \cdots & \mathrm{I}_{\ell_2} \\ \mathrm{I}_{\ell_2+1} & \mathrm{I}_{\ell_2+2} & \cdots & \mathrm{I}_{2\,\ell_2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{I}_{(\ell_1-1)\ell_2+1} & \mathrm{I}_{(\ell_1-1)\ell_2+2} & \cdots & \mathrm{I}_{\ell_1\ell_2} \end{pmatrix} \;=\; \begin{pmatrix} \mathrm{I}_{(1,1)} & \mathrm{I}_{(1,2)} & \cdots & \mathrm{I}_{(1,\ell_2)} \\ \mathrm{I}_{(2,1)} & \mathrm{I}_{(2,2)} & \cdots & \mathrm{I}_{(2,\ell_2)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{I}_{(\ell_1,1)} & \mathrm{I}_{(\ell_1,2)} & \cdots & \mathrm{I}_{(\ell_1,\ell_2)} \end{pmatrix}.$$

Using this convention, we can now describe the hybrid HIBE system as follows.

`Setup`(ℓ, ω)**:** For a HIBE of maximum depth ℓ, first determine ℓ_1 and ℓ_2 as above so that $\ell \le \ell_1 \cdot \ell_2$. Next, select a random generator g in $\mathbb{G}$, a random $\alpha \in \mathbb{Z}_p$, and set $g_1 = g^{\alpha}$. Then, pick random elements $g_2, f_1, \ldots, f_{\ell_1}, h_1, \ldots, h_{\ell_2} \in \mathbb{G}$. The public parameters *params* and the secret *master-key* are given by

$$\mathit{params} = (\ g,\ g_1, g_2,\ f_1, \ldots, f_{\ell_1},\ h_1, \ldots, h_{\ell_2}\ )\ , \qquad \mathit{master\text{-}key} = g_2^{\alpha}.$$

`KeyGen`$(d_{\mathsf{ID}|k-1}, \mathsf{ID})$**:** To generate private key d_{ID} for identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in (\mathbb{Z}_p^*)^k$ of depth

$(k_1, k_2) = k \leq \ell$, where $k_1 \leq \ell_1$ and $k_2 \leq \ell_2$, pick random $r_1, \ldots, r_{k_1} \in \mathbb{Z}_p$, and output

$$\begin{aligned} d_{\mathsf{ID}} \;=\; \Bigg(& g_2^{\alpha} \cdot \left(\prod_{i=1}^{k_1 - 1} \left(h_1^{\mathrm{I}_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}} \cdot f_i \right)^{r_i} \right) \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1} \right)^{r'_{k_1}}, \\ & g^{r_1}, \ \ldots, \ g^{r_{k_1-1}}, \ g^{r'_{k_1}}, \ h_{k_2+1}^{r'_{k_1}}, \ \ldots, \ h_{\ell_2}^{r'_{k_1}} \Bigg) \ \in \ \mathbb{G}^{1+k_1+\ell_2-k_2}. \end{aligned} \tag{6}$$

Note that the factors $(\ldots)^{r_i}$ under the $\prod$ sign contain ℓ_2 identity terms each, whereas the last factor $(\ldots)^{r_{k_1}}$ only has k_2 such terms. The size of d_{ID} grows with k_1 and shrinks with k_2; the private key thus becomes alternatively shorter and longer as the depth of ID increases, but never exceeds $\ell_1 + \ell_2$ elements of $\mathbb{G}$.

The private key for ID can be generated with a private key for $\mathsf{ID}_{|k-1} = (\mathrm{I}_1, \ldots, \mathrm{I}_{k-1}) \in (\mathbb{Z}_p^*)^{k-1}$ as required. Decompose k as (k_1, k_2) according to our convention. There are two cases:

1. If $k-1$ is written $(k_1, k_2 - 1)$, namely k and $k-1$ have the same row index k_1, then we know that the private key for $\mathsf{ID}_{|k-1}$ is of the form:

$$\begin{aligned} d_{\mathsf{ID}|k-1} = \Bigg(& g_2^{\alpha} \cdot \prod_{i=1}^{k_1-1} \left(h_1^{\mathrm{I}_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}} \cdot f_i \right)^{r_i} \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2-1}^{\mathrm{I}_{(k_1,k_2-1)}} \cdot f_{k_1} \right)^{r_{k_1}}, \ g^{r_1}, \\ & \ldots, \ g^{r_{k_1}}, h_{k_2}^{r_{k_1}}, \ \ldots, \ h_{\ell_2}^{r_{k_1}} \Bigg) \ = \ (a_0, b_1, \ldots, b_{k_1}, c_{k_2}, \ldots, c_{\ell_2}) \ \in \mathbb{G}^{2+k_1+\ell_2-k_2}. \end{aligned}$$

In this case, to generate d_{ID} from $d_{\mathsf{ID}|k-1}$, pick a random $r^* \in \mathbb{Z}_p$ and output

$$\begin{aligned} d_{\mathsf{ID}} = \Bigg(& a_0 \cdot c_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1} \right)^{r^*}, \ \ b_1, \ \ldots, \ b_{k_1-1}, \ b_{k_1} \cdot g^{r^*}, \\ & c_{k_2+1} \cdot h_{k_2+1}^{r^*}, \ \ldots, \ c_{\ell_2} \cdot h_{\ell_2}^{r^*} \Bigg) \ \in \mathbb{G}^{1+k_1+\ell_2-k_2}. \end{aligned}$$

This tuple is of the same form as Eq (6) where $r'_{k_1} = r_{k_1} + r^*$.

2. If the row indices differ, then necessarily $k - 1 = (k_1 - 1, \ell_2)$ and $k = (k_1, 1)$, and the private key for $\mathsf{ID}_{|k-1}$ must be of the form:

$$\begin{aligned} d_{\mathsf{ID}|k-1} &= \left(g_2^{\alpha} \cdot \prod_{i=1}^{k_1-1} \left(h_1^{\mathrm{I}_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}} \cdot f_i \right)^{r_i}, \ \ g^{r_1}, \ \ldots, \ g^{r_{k_1-1}} \right) \\ &= (a_0, b_1, \ldots, b_{k_1-1}) \ \in \mathbb{G}^{k_1}. \end{aligned}$$

In this case, to generate d_{ID} from $d_{\mathsf{ID}|k-1}$, pick a random $r \in \mathbb{Z}_p$ and output

$$d_{\mathsf{ID}} = \left(a_0 \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdot f_{k_1} \right)^{r}, \ \ b_1, \ \ldots, \ b_{k_1-1}, \ \ g^{r}, \ \ h_2^{r}, \ \ , \ldots, h_{\ell_2}^{r} \right) \ \in \mathbb{G}^{k_1+\ell_2}.$$

Again, this tuple conforms to Eq (6) in which r_{k_1} has been set to r.

$\texttt{Encrypt}(params, \mathsf{ID}, M)$: To encrypt a message $M \in \mathbb{G}_1$ under the public key $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in \mathbb{Z}_p^k$ where $k = (k_1, k_2)$, pick a random $s \in \mathbb{Z}_p$ and output

$$\mathsf{CT} = \Big(e(g_1, g_2)^s \cdot M, \quad g^s, \quad \big(h_1^{\mathrm{I}_{(1,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(1,\ell_2)}} \cdot f_1\big)^s, \ \ldots \ ,$$
$$\big(h_1^{\mathrm{I}_{(k_1-1,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(k_1-1,\ell_2)}} \cdot f_{k_1-1}\big)^s, \ \big(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1}\big)^s \Big) \ \in \mathbb{G}_1 \times \mathbb{G}^{1+k_1}.$$

$\texttt{Decrypt}(d_{\mathsf{ID}}, \mathsf{CT})$: Consider an identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$ with $k = (k_1, k_2)$. To decrypt a ciphertext $\mathsf{CT} = (A, B, C_1, \ldots, C_{k_1-1}, C_{k_1})$ using the private key $d_{\mathsf{ID}} = (a_0, b_1, \ldots, b_{k_1}, c_{k_2+1}, \ldots, c_{\ell_2})$, output

$$A \cdot \prod_{i=1}^{k_1} e(b_i, C_i) \Big/ e(B, a_0) = M.$$

Note that the private key components $c_{k_2+1}, \ldots, c_{\ell_2}$ are not used for decryption.

Complexity. It is easy to see that in a hierarchy of depth ℓ, private keys at all levels contain at most $\ell_1 + \ell_2$ elements of $\mathbb{G}$, while ciphertexts contain at most $1 + \ell_1$ elements of $\mathbb{G}$ and one element of $\mathbb{G}_1$. Encryption, decryption, and one-level-down key generation, all require $O(\ell_1 + \ell_2)$ operations, or $O(\sqrt{\ell})$ for the choice $\omega = 1/2$ as claimed. We note that the combination of having a selectable parameter ω together with the option of using an asymmetric bilinear group geared toward reducing the ciphertext or the private key size (described in Section 4.3), gives great flexibility toward achieving the optimal trade-off for a given application.

Security. We prove security based on the (ℓ_2)-wBDHI assumption. We note that for $\omega = 1/2$, security for a ℓ-level hierarchy is based on the $O(\sqrt{\ell})$-wBDHI assumption.

Theorem 4.1. *Let $\mathbb{G}$ be a bilinear group of prime order p. Consider a hybrid ℓ-HIBE system with identity hierarchy partitioned into ℓ_1 groups each of size ℓ_2. Suppose the Decision $(t, \epsilon, \ell_2 + 1)$-wBDHI assumption holds in $\mathbb{G}$. Then the hybrid ℓ-HIBE system is (t', q_S, ϵ)-selective identity, chosen plaintext* (IND-sID-CPA) *secure for arbitrary ℓ, q_S, and $t' < t - \Theta(\tau\, \ell\, q_S)$, where τ is the maximum time for an exponentiation in $\mathbb{G}$.*

The proof is similar to that for Theorem 3.1 and is given in Appendix B.

4.3 Asymmetric Bilinear Groups and MNT Curves.

It is often desirable to use bilinear maps $e : \mathbb{G} \times \mathbb{G}' \to \mathbb{G}_1$ where $\mathbb{G}$ and $\mathbb{G}'$ are distinct groups. Such maps let us take advantage of certain curves called MNT curves [MNT01]. Typically, elements of the group $\mathbb{G}$ tend to afford a particularly compact representation compared to elements of $\mathbb{G}'$. This property is used for constructing short signatures [BLS01, BB04b, BBS04]. For our system, we can use this property to shrink either the private keys or the ciphertexts.

We briefly describe how to rephrase the above HIBE in terms of asymmetric bilinear groups $e : \mathbb{G} \times \mathbb{G}' \to \mathbb{G}_1$. Recall that such groups are necessarily equipped with an efficiently computable homomorphism $\phi : \mathbb{G}' \to \mathbb{G}$. Thus, `Setup` picks all randomly selected group elements from $\mathbb{G}'$. Before these elements are used or disclosed, each element is either converted to a point in $\mathbb{G}$ using ϕ, or left alone in $\mathbb{G}'$, depending on whether this particular element is destined to appear in the first or second argument of the bilinear map $e(\cdot, \cdot)$. It is easy to see that there are no conflicts. There are two ways to perform this conversion: we can either send the private key or the ciphertext elements to $\mathbb{G}$; in one case we end up with smaller private keys, and in the other, smaller ciphertexts.

5 Applications

We now discuss applications of our compact HIBE system and its extensions.

5.1 Forward Secure Encryption

The main purpose of a forward secure encryption scheme is to guarantee that all messages encrypted before the secret key is compromised remain secret.

An elegant public key encryption scheme with forward security was proposed by Canetti, Halevi, and Katz (CHK) [CHK03]. Let $T = 2^t$ be the number of distinct time periods in the forward secure system. When implemented with previous HIBE systems [GS02, BB04a], the CHK framework results in ciphertexts of size $O(t)$ and private keys of size $O(t^2)$. Using public updateable storage, Canetti et al. reduce private key size to $O(t)$ without affecting ciphertext length — the idea is to encrypt the private key for time period i under the public key of time period $i-1$ and store the resulting ciphertext, of size $O(t^2)$, in public storage; consequently, only one HIBE private key of size $O(t)$ is kept in private storage.

Using the HIBE system of Section 3 in the CHK framework, we obtain a forward secure encryption scheme with $O(1)$ ciphertext size and decryption time — independent of the number of time periods. Private keys using our basic system are of size $O(t^2)$. Alternatively, using the hybrid HIBE of Section 4.2 in which we set $\omega = 1/2$, we obtain a forward secure encryption scheme with private key size $O(t^{3/2})$; in this case ciphertext size and decryption time become $O(\sqrt{t})$.

Following Canetti et al. [CHK03], we can store most of the private key in updateable public storage in order to lessen the private storage requirement. Applied to our basic forward secure system, using $O(t^2)$ public storage we can reduce the private key size to $O(t)$ while keeping the ciphertext size constant. Using the hybrid HIBE system (for $\omega = 1/2$), the private storage requirement can be similarly reduced to $O(\sqrt{t})$ at the cost of $O(t^{3/2})$ updateable public storage; ciphertext size in this case remains $O(\sqrt{t})$.

5.2 Forward Secure HIBE

Recently, a forward secure HIBE scheme was proposed by Yao et al. [YFDL04]. Their scheme essentially uses two HIBE hierarchies in the manner of Canetti et al. [CHK03] to obtain forward security together with the ability to derive subordinate keys. Their system has ciphertexts of size $O(\ell \cdot t)$ where ℓ is the depth of the identity hierarchy and $T = 2^t$ is the number of time periods. Indeed, they pose as an open problem if a forward secure HIBE scheme with "linear" complexity is possible.

Instantiating both hierarchies in their construction with our HIBE system immediately gives a forward secure HIBE scheme with ciphertexts of size $O(1)$, which resolves this question.

We also propose a more specific forward secure HIBE construction that achieves "linear" $O(\ell+t)$ size for all components, including private keys and public parameters (ciphertexts are no longer constant size in that construction). The construction is a hybrid between the HIBE given in Section 3 and the Boneh-Boyen HIBE from [BB04a]; it is described in detail in Appendix C.

5.3 Public Key NNL Broadcast Encryption

Broadcast encryption schemes, introduced by Fiat and Naor [FN93], are cryptosystems designed for the efficient broadcast of data to a dynamic group of users authorized to receive the data. Naor, Naor, and Lotspiech [NNL01] considered broadcast encryption in the stateless receiver setting; they provided a general "subset cover" framework for such broadcast encryption schemes and gave two instances of the framework — the Complete Subtree (CS) method and the more efficient Subset Difference (SD) method. Further improvements have been proposed such as the Layered Subset Difference (LSD) [HS02] and the Stratified Subset Difference (SSD) [GST04]. In the symmetric key setting, only a "center" that possesses the secret keys can broadcast to the users. In a public key broadcast encryption system, anyone is allowed to broadcast to selected subsets of users.

Using the HIBE framework, Dodis and Fazio [DF02] showed how to translate the SD and LSD methods to the public key setting. For N users and r revoked users, their SD and LSD constructions based on previous HIBE systems give ciphertexts of size $O(r \cdot \log N)$, which is no better than the basic CS method. Substituting the HIBE system of Section 3 restores the full benefits of both SD and LSD, which results in ciphertexts of size $O(r)$.

5.4 Encrypting to the Future

Mont et al. [MHS03] observed that an IBE system gives a mechanism for encrypting to the future using a trusted server. Let D be a certain date string. We view D as a public key in an IBE system. Every day, a trusted server publishes the private key corresponding to that day, which enables messages encrypted for that day to be decrypted. Methods for encrypting to the future without a trusted server were proposed by Rivest, Shamir, and Wagner [RSW96].

One problem with the IBE timelock mechanism is that after n days have passed, the server has to publish a bulletin board with n private keys on it (one private key for each day). The amount of data on the bulletin board can be greatly reduced by using the CHK forward secure encryption scheme *in reverse*. Suppose the CHK framework is set up for a total of T time periods (using a tree of depth $\log_2 T$). To encrypt a message for day $n < T$, use the CHK public key for time period $T - n$. Similarly, on day n the trusted server publishes the CHK private key corresponding to time period $T - n$. This single private key enables anyone to derive the private keys for CHK time periods $T - n, T - n + 1, \ldots, T$. Anyone can thus decrypt messages intended for days in the range $1, \ldots, n$.

Implementing this encoding using our $O(1)$ ciphertext HIBE, the trusted server on any day only needs to publish a single private key comprising $O(\log^2 T)$ group elements. Using the hybrid HIBE system of Section 4.2, the private key posted by the server is further reduced to $O(\log^{3/2} T)$ group elements for ciphertexts of size $O(\sqrt{\log T})$. These parameters are much better than the IBE based mechanism [MHS03], where the bulletin board contains as many as T group elements.

6 Conclusions and Open Problems

We presented a new HIBE system where the ciphertexts consist of three group elements and decryption only requires computing two bilinear maps, both of which are independent of the hierarchy depth. Encryption time is as efficient as other HIBE systems. For a hierarchy of depth ℓ, we proved security based on the (Weak) ℓ-BDHI assumption.

We discussed several applications of our system, including efficient forward secure encryption, an efficient public key version of the NNL broadcast encryption system, and an efficient mechanism for encrypting to the future. Our HIBE system allows for limited delegation and can be combined with the Boneh-Boyen HIBE to form a hybrid HIBE that has sublinear private key size.

We note that our selective-ID proof of security is tight. On the other hand, the proof of full security (either in the random oracle or standard model) discussed in Section 3.2 degrades exponentially in the hierarchy depth. The same is true for all existing HIBE systems. It is an open problem to construct a HIBE system where security does not degrade exponentially in the hierarchy depth.

Acknowledgments

In an earlier version of this paper [BBG05] the security theorem used the Bilinear Diffie-Hellman Exponent assumption, defined in Appendix A.1; we thank Nelly Fazio for pointing out that our proof of security holds, without any change, under the weaker BDHI assumption. The authors also thank Mihir Bellare for his helpful comments.

References

[BB04a] Dan Boneh and Xavier Boyen. Efficient selective-ID identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 223–38. Springer, 2004.

[BB04b] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Proceedings of Eurocrypt 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, 2005.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matt Franklin, editor, *Proceedings of Crypto 2004*, LNCS, pages 41–55. Springer, 2004.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Proceedings of Crypto 2001*, volume 2139 of *LNCS*, pages 213–29. Springer, 2001.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. Cryptology ePrint Archive, Report 2005/018, 2005.

[BK05] Dan Boneh and Jonathan Katz. Improved efficiency for CCA-secure cryptosystems built using identity based encryption. In *Proceedings of RSA-CT 2005*, 2005.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Proceedings of Asiacrypt 2001*, volume 2248 of *LNCS*, pages 514–32. Springer, 2001.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In Eli Biham, editor, *Proceedings of Eurocrypt 2003*, volume 2656 of *LNCS*. Springer, 2003.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *Proceedings of Eurocrypt 2004*, volume 3027 of *LNCS*, pages 207–22. Springer, 2004.

[DF02] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In Joan Feigenbaum, editor, *Proceedings of the Digital Rights Management Workshop 2002*, volume 2696 of *LNCS*, pages 61–80. Springer, 2002.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In *Proceedings of the Workshop on Theory and Practice in Public Key Cryptography 2005*, 2005.

[FN93] Amos Fiat and Moni Naor. Broadcast encryption. In Douglas Stinson, editor, *Proceedings of Crypto 1993*, volume 773 of *LNCS*, pages 480–91. Springer, 1993.

[GS02] Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Proceedings of Asiacrypt 2002*, volume 2501 of *LNCS*, pages 548–66, 2002.

[GST04] Michael Goodrich, Jonathan Sun, and Robert Tamassia. Efficient tree-based revocation in groups of low-state devices. In Matthew Franklin, editor, *Proceedings of Crypto 2004*, volume 3152 of *LNCS*, pages 511–27. Springer, 2004.

[HL02] Jeremy Horwitz and Ben Lynn. Towards hierarchical identity-based encryption. In Lars Knudsen, editor, *Proceedings of Eurocrypt 2002*, volume 2332 of *LNCS*, pages 466–81. Springer, 2002.

[HS02] Dani Halevy and Adi Shamir. The LSD broadcast encryption scheme. In Moti Yung, editor, *Proceedings of Crypto 2002*, volume 2442 of *LNCS*, pages 47–60, 2002.

[Jou00] Antoine Joux. A one round protocol for tripartite Diffie-Hellman. In Wieb Bosma, editor, *Proceedings of Algorithmic Number Theory Symposium IV*, volume 1838 of *LNCS*, pages 385–94. Springer, 2000.

[MHS03] Marco Casassa Mont, Keith Harrison, and Martin Sadler. The HP time vault service: exploiting IBE for timed release of confidential information. In *Proceedings of the International World Wide Web Conference 2003*, pages 160–69. ACM, 2003.

[MNT01] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New explicit conditions of elliptic curve traces for FR-reduction. *IEICE Trans. Fundamentals*, E84-A(5):1234–43, 2001.

[MSK02] Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. A new traitor tracing. *IEICE Transactions Fundamentals*, E85-A(2):481–84, 2002.

[NNL01] Dalit Naor, Moni Naor, and Jeff Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *Proceedings of Crypto 2001*, volume 2139 of *LNCS*, pages 41–62. Springer, 2001.

[RSW96] Ronald Rivest, Adi Shamir, and David Wagner. Time-lock puzzles and timed-release crypto. Technical Report MIT/LCS/TR-684, MIT Laboratory for Computer Science, 1996.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. Blakley and David Chaum, editors, *Proceedings of Crypto 1984*, volume 196 of *LNCS*, pages 47–53. Springer, 1984.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Proceedings of Eurocrypt 1997*, volume 1233 of *LNCS*, pages 256–66. Springer, 1997.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Proceedings of Eurocrypt 2005*, LNCS. Springer, 2005.

[YFDL04] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In Birgit Pfitzmann, editor, *Proceedings of the ACM Conference on Computer and Communications Security 2004*, pages 354–63, 2004.

A Diffie-Hellman Problems in Generic Bilinear Groups

Recently a number of Diffie-Hellman-type complexity assumptions in bilinear groups have been used to construct efficient cryptosystems. These include the Bilinear DH assumption (BDH) [BF01], the DH Inversion assumption (DHI) [BB04a], the Linear DH assumption [BBS04], and the BDHE assumption used in [BGW05], and others. To gain some confidence in these assumptions, one can prove computational lower bounds on the difficulty of breaking them in a generic bilinear group model [Sho97]. Rather than prove a separate lower bound for each assumption, we give a general lower bound that encompasses all the assumptions listed above. This "master" generic-group lower bound can be used to qualify other assumptions that may come up in future constructions. We emphesize, however, that lower bounds in generic groups do not imply a lower bound in any specific group.

The Strong Diffie-Hellman (SDH) assumption [BB04b] stands out from the list above and is not covered by the "master" argument. We briefly mention the reasons for this in Section A.4.

A.1 Bilinear Diffie-Hellman Exponent (BDHE) Problem

An early version of this paper [BBG05] introduced the BDHE assumption. The assumption is no longer used in the paper; we use the weaker BDHI assumption instead. Nevertheless, we briefly describe the BDHE assumption for completeness. We note that the BDHE assumption has been used to construct a broadcast encryption system where both ciphertexts and private keys are short [BGW05].

Let g and h be random generators in $\mathbb{G}$, and let α be random in $\mathbb{Z}_p^*$. The ℓ-BDHE problem in $\mathbb{G}$ is as follows:

$$\ell\text{-BDHE:} \qquad \text{given } \; g, h, \text{ and } g^{(\alpha^i)} \text{ for } i = 1, 2, \ldots, \ell-1, \ell+1, \ldots, 2\ell \qquad \text{output } \; e(g,h)^{(\alpha^\ell)}$$

Since $g^{(\alpha^\ell)}$ is missing from the list of powers, the bilinear map seems to be of no help in computing $e(g,h)^{(\alpha^\ell)}$. As a shorthand, let $y_i = g^{(\alpha^i)} \in \mathbb{G}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving the computational ℓ-BDHE problem in $\mathbb{G}$ if

$$\Pr\left[\mathcal{A}\big(g, h, y_1, \ldots, y_{\ell-1}, y_{\ell+1}, \ldots, y_{2\ell}\big) = e(g,h)^{(\alpha^\ell)}\right] \geq \epsilon,$$

where the probability is over the random choice of generators g, h in $\mathbb{G}$, the random choice of α in $\mathbb{Z}_p^*$, and the random bits used by $\mathcal{A}$. The decisional version of the ℓ-BDHE problem in $\mathbb{G}$ is defined in the usual manner.

For any ℓ, the $(\ell+1)$-BDHE problem is similar to the ℓ-wBDHI* problem described in Section 2.3, except that additional terms are given in which g is raised to higher powers of α beyond a "gap" at exponent α^ℓ (which is always omitted). The generic lower bound presented in the "master" argument below applies to the BDHE problem.

A.2 General Diffie-Hellman Exponent Problem

Let p be an integer prime and let s, n be positive integers. Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$ and let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. Thus, P and Q are just two ordered sets containing s multi-variate polynomials each. We write $P = (p_1, p_2, \ldots, p_s)$ and $Q = (q_1, q_2, \ldots, q_s)$. We require that the first components of P, Q satisfy $p_1 = q_1 = 1$; that is, the constant polynomials 1. For a set Ω, a function $h : \mathbb{F}_p \to \Omega$, and a vector $x_1, \ldots, x_n \in \mathbb{F}_p$, we write

$$h(P(x_1, \ldots, x_n)) = \big(h(p_1(x_1, \ldots, x_n)),\ \ldots,\ h(p_s(x_1, \ldots, x_n)) \big) \in \Omega^s.$$

We use similar notation for the s-tuple Q. Let $\mathbb{G}_0, \mathbb{G}_1$ be groups of order p and let $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ be a non-degenerate bilinear map. Let $g \in \mathbb{G}_0$ be a generator of $\mathbb{G}_0$ and set $g_1 = e(g, g) \in \mathbb{G}_1$.

We define the (P, Q, f)-Diffie-Hellman Problem in $\mathbb{G}$ as follows: Given the vector

$$H(x_1, \ldots, x_n) = \big(g^{P(x_1,\ldots,x_n)},\ g_1^{Q(x_1,\ldots,x_n)} \big) \in \mathbb{G}_0^s \times \mathbb{G}_1^s,$$

compute $g_1^{f(x_1,\ldots,x_n)} \in \mathbb{G}_1$.

To obtain the most general result, we study the decisional version of this problem. We say that an algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the Decision (P, Q, f)-Diffie-Hellman problem in $\mathbb{G}_0$ if

$$\left| \Pr\Big[\mathcal{B}(H(x_1, \ldots, x_n),\ g_1^{f(x_1,\ldots,x_n)}) = 0\Big] - \Pr\Big[\mathcal{B}(H(x_1, \ldots, x_n),\ T) = 0\Big] \right| > \epsilon$$

where the probability is over the random choice of generator $g \in \mathbb{G}_0$, the random choice of $x_1, \ldots, x_n$ in $\mathbb{F}_p$, the random choice of $T \in \mathbb{G}_1$, and the random bits consumed by $\mathcal{B}$.

Before we can state the lower bound on the decision problem above, we need the following natural definition.

Definition A.1. Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$. Write $P = (p_1, p_2, \ldots, p_s)$ and $Q = (q_1, q_2, \ldots, q_s)$ where $p_1 = q_1 = 1$. We say that a polynomial $f \in \mathbb{F}_p[X_1, \ldots, X_n]$ is dependent on the sets (P, Q) if there exist $s^2 + s$ constants $\{a_{i,j}\}_{i,j=1}^s$, $\{b_k\}_{k=1}^s$ such that

$$f = \sum_{i,j=1}^{s} a_{i,j} p_i p_j + \sum_{k=1}^{s} b_k q_k$$

We say that f is independent of (P, Q) if f is not dependent on (P, Q).

For a polynomial $f \in \mathbb{F}_p[X_1, \ldots, X_n]^s$, we let d_f denote the total degree of f. For a set $P \subseteq \mathbb{F}_p[X_1, \ldots, X_n]^s$ we let $d_P = \max\{d_f \mid f \in P\}$.

A.3 Complexity Lower Bound in Generic Bilinear Groups

We state the following lower bound in the framework of the generic group model [Sho97]. We consider two random encodings ξ_0, ξ_1 of the additive group $\mathbb{Z}_p^+$, i.e. injective maps $\xi_0, \xi_1 : \mathbb{Z}_p^+ \to \{0,1\}^m$. For $i = 0, 1$ we write $\mathbb{G}_i = \{\xi_i(x) \mid x \in \mathbb{Z}_p^+\}$. We are given oracles to compute the induced group action on $\mathbb{G}_0, \mathbb{G}_1$, and an oracle to compute a non-degenerate bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. We refer to $\mathbb{G}_0$ as a *generic* bilinear group. The following theorem gives a lower bound on the advantage of a generic algorithm in solving the decision (P, Q, f)-Diffie-Hellman problem. We emphasize, however, that a lower bound in generic groups does not imply a lower bound in any specific group.

Theorem A.2. *Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$ and let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. Let $d = \max(2d_P, d_Q, d_f)$. Let ξ_0, ξ_1 and $\mathbb{G}_0, \mathbb{G}_1$ be defined as above. If f is independent of (P, Q) then for any $\mathcal{A}$ that makes a total of at most q queries to the oracles computing the group operation in $\mathbb{G}_0, \mathbb{G}_1$ and the bilinear pairing $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ we have:*

$$\left| \Pr\left[\mathcal{A}\left(\begin{array}{c} p,\ \xi_0(P(x_1,\ldots,x_n)), \\ \xi_1(Q(x_1,\ldots,x_n)), \\ \xi_1(t_0),\ \xi_1(t_1) \end{array} \right) = b \ :\ \begin{array}{c} x_1,\ldots,x_n, y \xleftarrow{\mathrm{R}} \mathbb{F}_p, \\ b \xleftarrow{\mathrm{R}} \{0,1\}, \\ t_b \leftarrow f(x_1,\ldots,x_n), \\ t_{1-b} \leftarrow y \end{array} \right] - \frac{1}{2} \right| \ \le\ \frac{(q+2s+2)^2 \cdot d}{2p} \ .$$

Proof. Consider an algorithm $\mathcal{B}$ that plays the following game with $\mathcal{A}$. Algorithm $\mathcal{B}$ maintains two lists of pairs,

$$L_0 = \{(p_i, \xi_{0,i}) : i = 1, \ldots, \tau_0\} \ , \qquad L_1 = \{(q_i, \xi_{1,i}) : i = 1, \ldots, \tau_1\} \ ,$$

under the invariant that at step τ in the game, $\tau_0 + \tau_1 = \tau + 2s + 2$. Here, $p_i \in \mathbb{F}_p[X_1, \ldots, X_n]$ and $q_i \in \mathbb{F}_p[X_1, \ldots, X_n, Y_0, Y_1]$ are multi-variate polynomials. The $\xi_{\star,\star}$ are strings in $\{0,1\}^m$.

The lists are initialized at step $\tau = 0$ by initializing $\tau_0 = s$ and $\tau_1 = s + 2$. We set $p_1, \ldots, p_s$ in L_0 to be the polynomials in P and we set $q_1, \ldots, q_s$ in L_1 to be the polynomials in Q. We also set $q_{s+1} = Y_0$ and $q_{s+2} = Y_1$. Algorithm $\mathcal{B}$ complete the preparation of the lists L_0, L_1 by setting the ξ-strings associated with distinct polynomials to random strings in $\{0,1\}^m$. Overall, initially L_0 contains s entries and L_1 contains $s + 2$ entries.

We can assume that $\mathcal{A}$ makes oracle queries only on strings obtained from $\mathcal{B}$, since $\mathcal{B}$ can make the strings in $\mathbb{G}_0, \mathbb{G}_1$ arbitrarily hard to guess by increasing m. We note that $\mathcal{B}$ can easily determine the index i of any given string $\xi_{j,i}$ in L_j (ties are broken arbitrarily). $\mathcal{B}$ starts the game by providing $\mathcal{A}$ with the value of p and a tuple of strings

$$\{\xi_{0,i}\}_{i=1}^{s},\ \{\xi_{1,i}\}_{i=1}^{s+2},$$

meant to encode some tuple $\in \mathbb{G}_0^s \times \mathbb{G}_1^{s+2}$. Algorithm $\mathcal{B}$ responds to $\mathcal{A}$'s oracle queries as follows.

Group operation in $\mathbb{G}_0, \mathbb{G}_1$. A query in $\mathbb{G}_0$ consists of two operands $\xi_{0,i}, \xi_{0,j}$ with $1 \le i, j \le \tau_0$ and a selection bit indicating whether to multiply or divide the group elements. To answer, let $\tau_0' \leftarrow \tau_0 + 1$. Perform the polynomial addition or subtraction $p_{\tau_0'} \leftarrow p_i \pm p_j$ depending on whether multiplication or division is requested. If the result $p_{\tau_0'} = p_l$ for some $l \le \tau_0$, then set $\xi_{0,\tau_0'} \leftarrow \xi_{0,l}$; otherwise, set $\xi_{0,\tau_0'}$ to a new random string in $\{0,1\}^m \setminus \{\xi_{0,1}, \ldots, \xi_{0,\tau_0}\}$. Insert the pair $(p_{\tau_0'}, \xi_{0,\tau_0'})$ into the list L_0 and update the counter $\tau_0 \leftarrow \tau_0'$. Algorithm $\mathcal{B}$ replies to $\mathcal{A}$ with the string $\xi_{0,\tau_0'}$.

$\mathbb{G}_1$ queries are handled analogously, this time by working with the list L_1 and the counter τ_1.

Bilinear pairing. A query of this type consists of two operands $\xi_{0,i}, \xi_{0,j}$ with $1 \le i, j \le \tau_0$. To respond, $\mathcal{B}$ sets $\tau_1' \leftarrow \tau_1 + 1$, and performs the polynomial multiplication $r_{\tau_1'} \leftarrow p_i \cdot p_j$. If the result $q_{\tau_1'} = q_l$ for some $l \le \tau_1$, it assigns $\xi_{1,\tau_1'} \leftarrow \xi_{1,l}$; otherwise, it sets $\xi_{1,\tau_1'}$ to a fresh random string in $\{0,1\}^m \setminus \{\xi_{1,1}, \ldots, \xi_{1,\tau_1 - 1}\}$. Finally, it adds $(q_{\tau_1'}, \xi_{1,\tau_1'})$ to the list L_1, updates $\tau_1 \leftarrow \tau_1'$, and outputs $\xi_{1,\tau_1'}$ as answer to the query.

After at most q queries, $\mathcal{A}$ terminates and returns a guess $b' \in \{0,1\}$. At this point $\mathcal{B}$ chooses random $x_1, \ldots, x_n, y \xleftarrow{R} \mathbb{F}_p$ and $b \xleftarrow{R} \{0,1\}$. Let $y_b = f(x_1, \ldots, x_n)$ and $y_{1-b} = y$.

For $i = 1, \ldots, n$, we set $X_i = x_i$, $Y_0 = y_0$, and $Y_1 = y_1$. It follows that the simulation provided by $\mathcal{B}$ is perfect unless the chosen random values for the variables $X_1, \ldots, X_n, Y_0, Y_1$ result in an equality relation between intermediate values that is not an equality of polynomials. In other words, the simulation is perfect unless for some i, j one of the following holds:

1. $p_i(x_1, \ldots, x_n) - p_j(x_1, \ldots, x_n) = 0$, yet the polynomials p_i and p_j are not equal.
2. $q_i(x_1, \ldots, x_n, y_0, y_1) - q_j(x_1, \ldots, x_n, y_0, y_1) = 0$, yet the polynomials q_i and q_j are not equal.

Let fail be the event that one of these two conditions holds. When event fail occurs, then $\mathcal{B}$'s responses to $\mathcal{A}$'s queries deviate from the real oracles' responses when the input tuple is derived from the vector $(x_1, \ldots, x_n, y_0, y_1) \in \mathbb{F}_p^{n+2}$.

We first bound the probability that event fail occurs. We bound the probability in two steps. First, consider setting $Y_b = f(X_1, \ldots, X_n)$. We claim that this symbolic substitution does not create any new equalities between polynomials q_i, q_j. In other words, if $q_i - q_j \ne 0$ for all i, j before the substitution, then $q_i - q_j \ne 0$ also holds after we set $Y_b = f(X_1, \ldots, X_n)$. This statement follows because f is independent of (P, Q). Indeed, $q_i - q_j$ is a polynomial of the form

$$\sum_{k=1}^{s} \sum_{l=1}^{s} a_{k,l}\, p_k\, p_l + \sum_{u=1}^{s} b_u\, q_u + c\, Y_0 + d\, Y_1$$

for some constants $a_{k,l}, b_u, c, d \in \mathbb{F}_p$. If this polynomial is non-zero but setting $Y_b = f(X_1, \ldots, X_n)$ causes this polynomial to vanish, then f must be dependent on (P, Q).

We are now left with polynomials in $X_1, \ldots, X_n, Y_{1-b}$. We need to bound the probability that for some i, j we get $(p_i - p_j)(x_1, \ldots, x_n) = 0$ even though $p_i - p_j \ne 0$, or that $(q_i - q_j)(x_1, \ldots, x, y) = 0$ even though $q_i - q_j \ne 0$. By construction, the maximum total degree of these polynomials is $d = \max(2d_P, d_Q, d_f)$. Therefore, for a given i, j the probability that a random assignment to $X_1, \ldots, X_n, Y_{1-b} \xleftarrow{R} \mathbb{F}_p$ is a root of $q_i - q_j$ is at most d/p. The same holds for $p_i - p_j$. Since there are no more than $2\binom{q+2s+2}{2}$ such pairs (p_i, p_j) and (q_i, q_j) in total, we have that

$$\Pr[\mathsf{fail}] \le \binom{q+2s+2}{2} \frac{2d}{p} \le (q + 2s + 2)^2 d/p$$

If event fail does not occur, then $\mathcal{B}$'s simulation is perfect. Furthermore, in this case b is independent from algorithm $\mathcal{A}$'s view. Indeed b is only chosen after the simulation ends. Hence, $\Pr[b = b' | \neg\mathsf{fail}] = 1/2$. It now follows that

$$\begin{array}{rcccl} \Pr[b = b'] & \le & \Pr[b = b' | \neg\mathsf{fail}](1 - \Pr[\mathsf{fail}]) + \Pr[\mathsf{fail}] & = & 1/2 + \Pr[\mathsf{fail}]/2 \\ \Pr[b = b'] & \ge & \Pr[b = b' | \neg\mathsf{fail}](1 - \Pr[\mathsf{fail}]) & = & 1/2 - \Pr[\mathsf{fail}]/2 \end{array}$$

and hence $|\Pr[b = b'] - 1/2| \le \Pr[\mathsf{fail}]/2 \le (q + 2s + 2)^2 d/2p$ as required $\square$

Corollary A.3. *Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$ and let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. Let $d = \max(2d_P, d_Q, d_f)$. If f is independent of (P, Q) then any $\mathcal{A}$ that has advantage $1/2$ in solving the decision (P, Q, f)-Diffie-Hellman Problem in a generic bilinear group $\mathbb{G}$ must take time at least $\Omega(\sqrt{p/d} - s)$.*

Using Corollary A.3. We briefly show that many standard (decisional) assumptions follow from Corollary A.3.

- DDH in $\mathbb{G}_1$: set $P = (1), Q = (1, x, y), f = xy$.
- BDH in $\mathbb{G}_0$: set $P = (1, x, y, z), Q = (1), f = xyz$.
- ℓ-BDHI in $\mathbb{G}_0$: set $P = (1, x, x^2, \ldots, x^\ell), Q = (1), f = x^{2\ell+1}$.
- ℓ-BDHE in $\mathbb{G}_0$: set $P = (1, y, x, x^2, \ldots, x^{\ell-1}, x^{\ell+1}, \ldots, x^{2\ell}), Q = (1), f = x^\ell y$.

Extensions. To take advantage of certain families of elliptic curves, such as the so-called MNT curves [MNT01], one often uses a more general bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_1 \to \mathbb{G}_2$ where the groups $\mathbb{G}_0$ and $\mathbb{G}_1$ are not necessarily the same. To accurately model the algebraic structure of these groups, we let $\psi : \mathbb{G}_1 \to \mathbb{G}_0$ be an efficiently computable isomorphism (on MNT curves, such an isomorphism is given by the trace map). We briefly state a similar result for these more general maps. We first generalize the definition of independence.

Definition A.4. Let $P, Q, R \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be three s-tuples of n-variate polynomials over $\mathbb{F}_p$. Write $P = (p_1, p_2, \ldots, p_s)$, $Q = (q_1, q_2, \ldots, q_s)$, and $R = (r_1, r_2, \ldots, r_s)$ where $p_1 = q_1 = r_1 = 1$. We say that a polynomial $f \in \mathbb{F}_p[X_1, \ldots, X_n]$ is dependent on the sets (P, Q, R) if there exist $2s^2 + s$ constants $\{a_{i,j}\}_{i,j=1}^s$, $\{b_{i,j}\}_{i,j=1}^s$, $\{c_k\}_{k=1}^s$ such that

$$f = \sum_{i=1}^{s} \sum_{j=1}^{s} a_{i,j}\, p_i\, q_j + \sum_{i=1}^{s} \sum_{j=1}^{s} b_{i,j} q_i\, q_j + \sum_{k=1}^{s} c_k\, r_k$$

We say that f is independent of (P, Q, R) if f is not dependent on (P, Q, R).

Given $P, Q, R \in \mathbb{F}_p[X_1, \ldots, X_n]^s$, and generators $g_0 \in \mathbb{G}_0,\ g_1 \in \mathbb{G}_1,\ g_2 \in \mathbb{G}_2$, we define the vector

$$H(x_1, \ldots, x_n) = \left(\, g_0^{P(x_1,\ldots,x_n)},\ g_1^{Q(x_1,\ldots,x_n)},\ g_2^{R(x_1,\ldots,x_n)} \,\right) \in \mathbb{G}_0^s \times \mathbb{G}_1^s \times \mathbb{G}_2^s$$

We say that an algorithm $\mathcal{B}$ that outputs $b \in \{0, 1\}$ has advantage ϵ in solving the Decision (P, Q, R, f)-Diffie-Hellman problem in $(\mathbb{G}_0, \mathbb{G}_1)$ if

$$\left| \Pr\left[\mathcal{B}(H(x_1, \ldots, x_n),\ g_1^{f(x_1,\ldots,x_n)}) = 0\right] - \Pr\left[\mathcal{B}(H(x_1, \ldots, x_n),\ T) = 0\right] \right| > \epsilon$$

where the probability is over the random choice of generators $g_0 \in \mathbb{G}_0,\ g_1 \in \mathbb{G}_1,\ g_2 \in \mathbb{G}_2$, the random choice of $x_1, \ldots, x_n$ in $\mathbb{F}_p$, the random choice of $T \in \mathbb{G}_2$, and the random bits consumed by $\mathcal{B}$. An identical proof to that of Theorem A.2 gives the following theorem.

Theorem A.5. *Let $P, Q, R \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be three s-tuples of n-variate polynomials over $\mathbb{F}_p$ and let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. Let $d = \max(d_P + d_Q, 2d_Q, d_R, d_f)$. If f is independent of (P, Q, R) then any $\mathcal{A}$ that has advantage $1/2$ in solving the decision (P, Q, R, f)-Diffie-Hellman Problem in a generic bilinear group $(\mathbb{G}_0, \mathbb{G}_1)$ must take time at least $\Omega(\sqrt{p/d} - s)$.*

A.4 The Case of the Strong Diffie-Hellman Assumption

As mentioned previously, the Strong Diffie-Hellman (SDH) assumption [BB04b] is not covered by the General Diffie-Hellman Exponent framework. A slightly simplified version of the ℓ-SDH problem can be stated as follows:

Given elements $g, g^{\alpha}, \ldots, g^{(\alpha^{\ell})}$ in $\mathbb{G}$, find a pair $(c, g^{1/(\alpha+c)})$ such that $c \neq 0 \bmod p$.

A closely related problem is to find $g^{1/(\alpha+c)}$ for some *given* value of $c \neq 0$. The two cases differ as follows:

- When c is fixed, the task is equivalent to the DHI problem [BB04a, DY05, MSK02], and is thus covered by the results of the previous section. We note that any hardness result on the *bilinear* DHI assumption immediately implies the hardness of the DHI assumption when the adversary no longer has access to the bilinear map oracle.
- When c is allowed to vary, as in the SDH problem [BB04b], the General Diffie-Hellman Exponent framework no longer applies because the problem now admits a large number of valid, distinct solutions (one for each c). In this respect, the SDH assumption is fundamentally different from DHI. We note that Boneh and Boyen [BB04b] give a direct proof of hardness of SDH in the generic group model.

B Security Proof for the Hybrid System of Section 4.2

In this section, we give a proof of Theorem 4.1, which is restated below for convenience.

Theorem B.1. *Let $\mathbb{G}$ be a bilinear group of prime order p. Consider a hybrid ℓ-HIBE system with identity hierarchy partitioned into ℓ_1 groups each of size ℓ_2. Suppose the Decision (t, ϵ, ℓ_2)-wBDHI assumption holds in $\mathbb{G}$. Then the hybrid ℓ-HIBE system is (t', q_S, ϵ)-selective identity, chosen plaintext* (IND-sID-CPA) *secure for arbitrary ℓ, q_S, and $t' < t - \Theta(\tau\, \ell\, q_S)$, where τ is the maximum time for an exponentiation in $\mathbb{G}$.*

Proof. Suppose $\mathcal{A}$ has advantage ϵ in attacking the hybrid ℓ-HIBE system. Using $\mathcal{A}$, we build an algorithm $\mathcal{B}$ that solves the Decision (ℓ_2)-wBDHI* problem in $\mathbb{G}$.

For a generator $g \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p$, let $y_i = g^{(\alpha^i)} \in \mathbb{G}$. Algorithm $\mathcal{B}$ is given as input a random tuple $(g, h, y_1, \ldots, y_{\ell_2}, T)$ that is either sampled from $\mathcal{P}_{wBDHI^*}$ (where $T = e(g,h)^{(\alpha^{\ell_2+1})}$) or from $\mathcal{R}_{wBDHI^*}$ (where T is uniform and independent in $\mathbb{G}_1^*$). Algorithm $\mathcal{B}$'s goal is to output 1 when the input tuple is sampled from $\mathcal{P}_{wBDHI^*}$ and 0 otherwise. Algorithm $\mathcal{B}$ works by interacting with $\mathcal{A}$ in a selective identity game as follows:

Initialization. The selective identity game begins with $\mathcal{A}$ first outputting an identity $\mathsf{ID}^* = (\mathrm{I}_1^*, \ldots, \mathrm{I}_m^*) \in \mathbb{Z}_p^m$ of depth $m \leq \ell$ that it intends to attack. If $m < \ell$ then $\mathcal{B}$ pads ID^* with $\ell - m$ zeroes on the right to make ID^* is a vector of length ℓ. From here on we assume that ID^* is a vector of length ℓ. Following our convention, we write ID^* as a pair (ℓ, I^*) where the matrix $\mathrm{I}^* \in \mathbb{Z}_p^{\ell_1 \times \ell_2}$ is filled using the elements $\mathrm{I}_1^*, \ldots, \mathrm{I}_\ell^*$.

Setup. To generate the system parameters, algorithm $\mathcal{B}$ picks a random γ in $\mathbb{Z}_p$ and sets $g_1 = y_1 = g^{\alpha}$ and $g_2 = y_{\ell_2} \cdot g^{\gamma} = g^{\gamma + (\alpha^{\ell_2})}$.

Next, $\mathcal{B}$ picks random $\gamma_1, \ldots, \gamma_{\ell_2}$ in $\mathbb{Z}_p$ and sets $h_i = g^{\gamma_i} / y_{\ell_2 - i + 1}$ for $i = 1, \ldots, \ell_2$.

Algorithm $\mathcal{B}$ also picks random $\delta_1, \ldots, \delta_{\ell_1}$ in $\mathbb{Z}_p$ and sets $f_i = g^{\delta_i} \cdot \prod_{j=1}^{\ell_2} (y_{\ell_2 - j + 1})^{I^*_{(i,j)}}$ for $i = 1, \ldots, \ell_1$.

Finally, $\mathcal{B}$ gives $\mathcal{A}$ the system parameters $params = (g, g_1, g_2, f_1, \ldots, f_{\ell_1}, h_1, \ldots, h_{\ell_2})$. Observe that all these values are distributed uniformly and independently in $\mathbb{G}$ as required.

The master key g_4 corresponding to these system parameters is $g_4 = g^{\alpha(\alpha^{\ell_2}+\gamma)} = y_{\ell_2+1} y_1^{\gamma}$, which is unknown to $\mathcal{B}$ since $\mathcal{B}$ does not have y_{ℓ_2+1}.

Phase 1. $\mathcal{A}$ issues up to q_S private key queries. Consider a query for the private key corresponding to $\mathsf{ID} = (\mathrm{I}_1, \ldots, I_u) \in \mathbb{Z}_p^u$ where $u \leq \ell$. The only restriction is that ID is not a prefix of ID^*. This restriction ensures that there exists a $k \in \{1, \ldots, u\}$ such that $\mathrm{I}_k \neq \mathrm{I}_k^*$ (otherwise, ID would be a prefix of ID^*); we set k such that it is the smallest such index. To respond to the query, algorithm $\mathcal{B}$ first derives a private key for the identity $\mathsf{ID}_k = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$ from which it then constructs a private key for the requested identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k, \ldots, \mathrm{I}_u)$.

As per our convention, we write ID_k as a pair (k, I) where the matrix $\mathrm{I} \in \mathbb{Z}_p^{\ell_1 \times \ell_2}$ is filled using the elements $\mathrm{I}_1, \ldots, \mathrm{I}_k$. Recall that our convention allows for decomposing the depth index k into a row-column pair $(k_1, k_2) = k$.

To generate the private key for the identity ID_k at depth $k = (k_1, k_2)$ where $k_1 \leq \ell_1$ and $k_2 \leq \ell_2$, algorithm $\mathcal{B}$ first picks random $r_1, \ldots, r_{k_1-1}, \tilde{r}_{k_1}$ in $\mathbb{Z}_p$. We pose $r_{k_1} = \frac{\alpha^{k_2}}{(I_{(k_1,k_2)} - I^*_{(k_1,k_2)})} + \tilde{r}_{k_1} \in \mathbb{Z}_p$. Next, $\mathcal{B}$ generates the private key

$$
\begin{aligned}
d_{\mathsf{ID}_k} \;=\; \Bigg(& g_4 \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1}\right)^{r_{k_1}} \cdot \left(\prod_{i=1}^{k_1-1} \left(h_1^{\mathrm{I}_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}} \cdot f_i\right)^{r_i} \right), \\
& g^{r_1}, \; \ldots, \; g^{r_{k_1-1}}, \; g^{r_{k_1}}, \; h_{k_2+1}^{r_{k_1}}, \; \ldots, \; h_{\ell_2}^{r_{k_1}} \Bigg) \quad \in \mathbb{G}^{1+k_1+\ell_2-k_2}.
\end{aligned}
\tag{7}
$$

which is a properly distributed private key for the identity ID_k. We show that $\mathcal{B}$ can compute all elements of this private key given the values at its disposal. We use the fact that $y_i^{(\alpha^j)} = y_{i+j}$ for any i, j.

We begin by showing how to generate the first component of the private key. Observe that

$$
\begin{aligned}
& \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1}\right)^{r_{k_1}} \\
&= \left(g^{r_{k_1} \cdot (\delta_{k_1} + \sum_{i=1}^{k_2} \mathrm{I}_{(k_1,i)} \gamma_i)} \cdot \prod_{i=1}^{k_2-1} y_{\ell_2-i+1}^{r_{k_1} \cdot (\mathrm{I}^*_{(k_1,i)} - \mathrm{I}_{(k_1,i)})} \cdot y_{\ell_2-k_2+1}^{r_{k_1} \cdot (\mathrm{I}^*_{(k_1,k_2)} - \mathrm{I}_{(k_1,k_2)})} \cdot \prod_{i=k_2+1}^{\ell_2} y_{\ell_2-i+1}^{r_{k_1} \cdot \mathrm{I}^*_{(k_1,i)}} \right).
\end{aligned}
\tag{8}
$$

Let Z denote the product of the first, second, and fourth terms. That is,

$$
Z = \left(g^{r_{k_1} \cdot (\delta_{k_1} + \sum_{i=1}^{k_2} \mathrm{I}_{(k_1,i)} \gamma_i)} \right) \cdot \underbrace{\prod_{i=1}^{k_2-1} y_{\ell_2-i+1}^{r_{k_1} \cdot (\mathrm{I}^*_{(k_1,i)} - \mathrm{I}_{(k_1,i)})}}_{=1} \cdot \prod_{i=k_2+1}^{\ell_2} y_{\ell_2-i+1}^{r_{k_1} \cdot \mathrm{I}^*_{(k_1,i)}}.
$$

Note that the second term in Z equals 1 because $\mathrm{I}_i = \mathrm{I}_i^*$ for all $i < k$ (namely, $\mathrm{I}_{(i,j)} = \mathrm{I}^*_{(i,j)}$ where both $i \leq k_1$ and $j < k_2$ hold). One can verify that $\mathcal{B}$ can compute all the terms in Z given the values at its disposal. On the other hand, $\mathcal{B}$ cannot compute the third term in Eq (8) by itself since it requires knowledge of $y_{\ell_2+1} = y_{\ell_2-k_2+1}^{\alpha^{k_2}}$. We, however, observe that

$$
y_{\ell_2-k_2+1}^{r_{k_1} \cdot (\mathrm{I}^*_{(k_1,k_2)} - \mathrm{I}_{(k_1,k_2)})} = y_{\ell_2-k_2+1}^{\tilde{r}_{k_1} \cdot (\mathrm{I}^*_{(k_1,k_2)} - \mathrm{I}_{(k_1,k_2)})} \big/ \, y_{\ell_2+1} = y_{\ell_2-k_2+1}^{\tilde{r}_{k_1} \cdot (\mathrm{I}^*_{(k_1,k_2)} - \mathrm{I}_{(k_1,k_2)})} \big/ \, (g_4 y_1^{-\gamma}).
$$

Hence, the product of the first two terms in the first component (7) in the private key is:

$$g_4 \cdot \left(h_1^{\mathrm{I}_{(k_1,1)}} \cdots h_{k_2}^{\mathrm{I}_{(k_1,k_2)}} \cdot f_{k_1}\right)^{r_{k_1}} = g_4 \cdot Z \cdot y_{\ell_2-k_2+1}^{\tilde{r}_{k_1}\cdot(\mathrm{I}^*_{(k_1,k_2)}-\mathrm{I}_{(k_1,k_2)})} \Big/ (g_4 y_1^{-\gamma})$$
$$= Z \cdot y_{\ell_2-k_2+1}^{\tilde{r}_{k_1}\cdot(\mathrm{I}^*_{(k_1,k_2)}-\mathrm{I}_{(k_1,k_2)})} \cdot y_1^{\gamma}.$$

Since g_4 cancels out from the expression on the right, all the terms in that expression are known to $\mathcal{B}$. Thus, $\mathcal{B}$ can compute the product of the first two terms in the first component of the private key (7). To conclude, note that the third term

$$\prod_{i=1}^{k_1-1} \left(h_1^{\mathrm{I}_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}_{(i,\ell_2)}} \cdot f_i\right)^{r_i} = \prod_{i=1}^{k_1-1} \left(g^{r_i\cdot(\delta_i+\sum_{j=1}^{\ell_2} \mathrm{I}_{(i,j)}\gamma_i)} \cdot \prod_{j=1}^{\ell_2} y_{\ell_2-j+1}^{r_i\cdot(\mathrm{I}^*_{(i,j)}-\mathrm{I}_{(i,j)})}\right)$$

has a similar form to Z and can be computed by $\mathcal{B}$ given the values at its disposal. Therefore, we have shown that $\mathcal{B}$ can compute the first component of the private key (7).

The components $g^{r_1}, \ldots, g^{r_{k_1-1}}$ are easily computed by raising g to the powers of $r_1, \ldots, r_{k_1-1}$, all of which $\mathcal{B}$ knows. The component $g^{r_{k_1}}$ is simply $y_{k_2}^{1/(I_{(k_1,k_2)}-I^*_{(k_1,k_2)})} \cdot g^{\tilde{r}_{k_1}}$, which $\mathcal{B}$ can also compute. Finally, observe that

$$h_{k_2+i}^{r_{k_1}} = \left(g^{\gamma_{k_2+i}} \big/ y_{\ell_2-k_2-i+1}\right)^{\frac{\alpha^{k_2}}{(I_{(k_1,k_2)}-I^*_{(k_1,k_2)})}+\tilde{r}_{k_1}} = \left(y_{k_2}^{\gamma_{k_2+i}} \big/ y_{\ell_2-i+1}\right)^{\frac{1}{(I_{(k_1,k_2)}-I^*_{(k_1,k_2)})}+\tilde{r}_{k_1}},$$

which $\mathcal{B}$ can compute for all $i = 1, \ldots, \ell_2 - k_2$ because there are no y_{ℓ_2+1} terms.

Thus, $\mathcal{B}$ can derive a valid private key for ID_k. Algorithm $\mathcal{B}$ derives a private key for the requested ID from this private key and gives $\mathcal{A}$ the result.

Challenge. When $\mathcal{A}$ decides that Phase 1 is over, it outputs two messages $M_0, M_1 \in \mathbb{G}_1$ on which it wishes to be challenged. Algorithm $\mathcal{B}$ picks a random bit $b \in \{0,1\}$ and responds with the challenge ciphertext

$$\mathsf{CT} = \left(M_b \cdot T \cdot e(y_1, h^{\gamma}),\ \ h,\ \ h^{\delta_1+\sum_{j=1}^{\ell_2} \mathrm{I}^*_{(1,j)}\gamma_j},\ h^{\delta_2+\sum_{j=1}^{\ell_2} \mathrm{I}^*_{(2,j)}\gamma_j},\ \ldots,\ h^{\delta_{\ell_1}+\sum_{j=1}^{\ell_2} \mathrm{I}^*_{(\ell_1,j)}\gamma_j}\right)$$

where h and T are from the input tuple given to $\mathcal{B}$. First note that if $h = g^c$ (for some unknown c in $\mathbb{Z}_p$) then

$$h^{\delta_i+\sum_{j=1}^{\ell_2} \mathrm{I}^*_{(i,j)}\gamma_j} = \left(h_1^{\mathrm{I}^*_{(i,1)}} \cdots h_{\ell_2}^{\mathrm{I}^*_{(i,\ell_2)}} \cdot f_i\right)^c$$

Therefore, if $T = e(g,h)^{(\alpha^{\ell_2+1})}$, (i.e., when the input tuple is sampled from $\mathcal{P}_{wBDHI^*}$) then the challenge ciphertext is:

$$\mathsf{CT} = \left(M_b \cdot e(g_1, g_2)^c,\quad g^c,\quad \left(h_1^{\mathrm{I}^*_{(1,1)}} \cdots h_{\ell_2}^{\mathrm{I}^*_{(1,\ell_2)}} \cdot f_1\right)^c, \ldots, \left(h_1^{\mathrm{I}^*_{(\ell_1,1)}} \cdots h_{\ell_2}^{\mathrm{I}^*_{(\ell_1,\ell_2)}} \cdot f_{\ell_1}\right)^c\right)$$

which is a valid encryption of M_b under the public key $\mathsf{ID}^* = (\mathrm{I}_1^*, \ldots, \mathrm{I}_\ell^*)$. On the other hand, when T is uniform and independent in $\mathbb{G}_1^*$ (when the input tuple is sampled from $\mathcal{R}_{wBDHI^*}$), then CT is independent of b in the adversary's view.

Phase 2. $\mathcal{A}$ continues to issue queries not issued in Phase 1. Algorithm $\mathcal{B}$ responds as before.

Guess. Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. Algorithm $\mathcal{B}$ concludes its own game by outputting a guess as follows. If $b = b'$ then $\mathcal{B}$ outputs 1 meaning $T = e(g,h)^{(\alpha^{\ell_2+1})}$. Otherwise, it outputs 0 meaning T is random in $\mathbb{G}_1$.

When the input tuple is sampled from $\mathcal{P}_{wBDHI^*}$ (where $T = e(g,h)^{(\alpha^{\ell_2+1})}$), then $\mathcal{A}$'s view is identical to its view in a real attack game and therefore $\mathcal{A}$ satisfies $|\Pr[b = b'] - 1/2| \geq \epsilon$. When the input tuple is sampled from $\mathcal{R}_{wBDHI^*}$ (where T is uniform in $\mathbb{G}_1^*$) then $\Pr[b = b'] = 1/2$. Therefore, with g, h uniform in $\mathbb{G}^*$, α uniform in $\mathbb{Z}_p$, and T uniform in $\mathbb{G}_1$ we have that

$$\left| \Pr\left[\mathcal{B}\big(g,h,\vec{y}_{g,\alpha,\ell_2}, e(g,h)^{(\alpha^{\ell_2+1})}\big) = 0\right] - \Pr\left[\mathcal{B}\big(g,h,\vec{y}_{g,\alpha,\ell_2}, T\big) = 0\right] \right| \quad \geq \quad |(1/2 \pm \epsilon) - 1/2| \; = \; \epsilon$$

as required, which completes the proof of the theorem. $\square$

C An fs-HIBE Linear in All Components

Using an idea related to the hybrid HIBE presented in Section 4.2, we now construct a forward secure HIBE system with a different complexity trade-off than that of Section 5.2. In the forthcoming scheme, ciphertexts, private keys, and public parameters all have "linear" $O(\ell + \log T)$ complexity, where ℓ is the depth of the identity hierarchy and T is the number of time slices for purposes of forward security. The scheme supports all the requirements set forth by Yao et al. [YFDL04], and in particular the "*dynamic-join*" ability for any identity at any level to generate private keys for any subordinate identities at any time.

The proposed fs-HIBE scheme, like the aforementioned hybrid HIBE, takes advantage of the complementarity and orthogonality of our basic HIBE (call it BBG) and that of Boneh and Boyen (BB). The main difference with the hybrid HIBE is that, here, each subsystem independently contributes its own hierarchy (time or identity) — something that Yao et al. [YFDL04] could not do based solely on the Gentry-Silverberg HIBE — so there is no need to consider a cross-product hierarchy. This fact allows us to achieve a complexity of $O(\ell + \log T)$ — rather than $O(\ell \cdot \log T)$ — for *all* components.

Consider a forward secure HIBE setting with $T = 2^t$ time periods and ℓ levels in the ID hierarchy. We construct such a fs-HIBE based on a t-level BB scheme for the time stamps and an ℓ-level BBG scheme for the identity hierarchy. Informally, each fs-HIBE private key consists of t private BB subkeys corresponding to selected nodes in a "time tree" à la Canetti et al. [CHK03]. Each subkey contains between 3 and $t+2$ elements, but because BB leaves most private key elements unchanged from parent to child, the whole set contains only about $2t$ elements (instead of the $t^2/2$ one would have expected). Then, viewing each BB subkey as a black box master key for a BBG system (exploiting the strong algebraic similarities between both systems), we use BBG to derive from this master key a subkey for the designated identity. Again, this adds up to ℓ elements to the subkey, but because the same ℓ elements can be shared among all t subkeys, the entire identity hierarchy ends up costing only a mere ℓ extra terms (instead of $t\,\ell$ as one would have expected). We end up with a total private key size $\approx 2t + \ell = O(\ell + \log T)$, a public key size $\approx t + \ell = O(\ell + \log T)$, and a ciphertext size $\approx t = O(\log T)$, which notably remains independent of the ID hierarchy depth.

C.1 Linear fs-HIBE Construction

As before, we assume a bilinear group $\mathbb{G}$ and a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$, where $\mathbb{G}$ and $\mathbb{G}_1$ have prime order p. Let $t = \log T$ be the depth of the time tree and ℓ be the depth of the ID hierarchy. As usual we let $\mathsf{ID} = (\mathrm{I}_1, \dots, \mathrm{I}_k) \in \mathbb{Z}_p^k$ be an identity of depth $k \le \ell$. The linear fs-HIBE system is described as follows.

$\texttt{Setup}(t, \ell)$**:** To generate system parameters for an fs-HIBE of maximum depth ℓ and a maximum of 2^t time periods: select a random generator g in $\mathbb{G}$, a random $\alpha \in \mathbb{Z}_p$, and set $g_1 = g^\alpha$; then, pick random elements $g_2, g_3, f_1, \dots, f_t, h_1, \dots, h_\ell \in \mathbb{G}$, and set $g_4 = g_2^\alpha$. The public parameters *params* and the secret *master-key* are given by

$$\textit{params} = (\ g,\ g_1, g_2, g_3,\ f_1, \dots, f_t,\ h_1, \dots, h_\ell\ )\ , \qquad \textit{master-key} = g_4 = g_2^\alpha .$$

$\texttt{KeyGen}(d_{\mathsf{ID}|k-1}, \mathsf{ID})$**:** To generate the private key $d_{\mathsf{ID}}^{(\tau)}$ for an identity $\mathsf{ID} = (\mathrm{I}_1, \dots, \mathrm{I}_k) \in \mathbb{Z}_p^k$ of depth $k \le \ell$ for the time period $\tau < 2^t$ with binary representation $\tau = \langle \tau_{t-1} | \cdots | \tau_0 \rangle \in \{0,1\}^t$, pick random $r_1, \dots, r_t, r'_1, \dots, r'_t, r \in \mathbb{Z}_p$, and compute

$$y \ = \ g_2^\alpha \cdot \left(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3\right)^r ,$$

to produce the output $d_{\mathsf{ID}} \in \mathbb{G}^{2+t+2\bar{t}+(\ell-k)}$ (where $\bar{t} = |\{\tau_i = 0 : 0 \le i < t\}|$), given by

$$d_{\mathsf{ID}}^{(\tau)} \ = \ \left(\begin{array}{ll} y \cdot \left(g_1^{\tau_{t-1}} \cdot f_1\right)^{r_1} \cdot \left(g_1^{\tau_{t-2}} \cdot f_2\right)^{r_2} \cdots \left(g_1^{\tau_1} \cdot f_{t-1}\right)^{r_{t-1}} \cdot \left(g_1^{\tau_0} \cdot f_t\right)^{r_t}\ , & \\ g^r , & \\ g^{r_1},\ g^{r_2},\ \dots,\ g^{r_{t-1}},\ g^{r_t}, & \\ y \cdot \left(g_1^{\tau_{t-1}} \cdot f_1\right)^{r_1} \cdot \left(g_1^{\tau_{t-2}} \cdot f_2\right)^{r_2} \cdots \left(g_1^{\tau_1} \cdot f_{t-1}\right)^{r_{t-1}} \cdot \left(g_1 \cdot f_t\right)^{r'_t}\ , & \text{if } \tau_0 = 0 \\ y \cdot \left(g_1^{\tau_{t-1}} \cdot f_1\right)^{r_1} \cdot \left(g_1^{\tau_{t-2}} \cdot f_2\right)^{r_2} \cdots \left(g_1 \cdot f_{t-1}\right)^{r'_{t-1}}\ , & \text{if } \tau_1 = 0 \\ \qquad \vdots & \quad \vdots \\ y \cdot \left(g_1^{\tau_{t-1}} \cdot f_1\right)^{r_1} \cdot \left(g_1 \cdot f_2\right)^{r'_2}\ , & \text{if } \tau_{t-2} = 0 \\ y \cdot \left(g_1 \cdot f_1\right)^{r'_1}\ , & \text{if } \tau_{t-1} = 0 \\ g^{r'_1},\ g^{r'_2},\ \dots,\ g^{r'_{t-1}},\ g^{r'_t}, \qquad \text{restricted to the } g^{r'_i} \text{ such that } \tau_{t-i} = 0 & \\ h_{k+1}^r,\ h_{k+2}^r,\ \dots,\ h_{\ell-1}^r,\ h_\ell^r & \end{array} \right) .$$

Observe that the private key for ID at time interval τ can be generated from a private key for $\mathsf{ID}_{|k-1} = (\mathrm{I}_1, \dots, \mathrm{I}_{k-1}) \in \mathbb{Z}_p^{k-1}$ with time index $\tau' \le \tau$, as required. We separately show how to derive $d_{\mathsf{ID}}^{(\tau)}$ respectively from $d_{\mathsf{ID}|k-1}^{(\tau)}$ and from $d_{\mathsf{ID}}^{(\tau')}$; the general case can be obtained by composing the two types of derivations in either order. The two cases are as follows.

1. $[d_{\mathsf{ID}|k-1}^{(\tau)} \to d_{\mathsf{ID}}^{(\tau)}]$. The private key for $\mathsf{ID}_{|k-1}$ with time index τ can be written as

$$d_{\mathsf{ID}|k-1}^{(\tau)} \ = \ \left(\begin{array}{ll} y' \cdot a'_0\ , & \\ z'\ , & \\ b_1,\ \dots,\ b_t, & \\ y' \cdot c'_t,\ \dots,\ y' \cdot c'_1, & \text{where } y' \cdot c'_i = \perp \text{ unless } \tau_{t-i} = 0 \\ d_1,\ \dots,\ d_t, & \text{where } d_i = \perp \text{ unless } \tau_{t-i} = 0 \\ h_k^r,\ \dots,\ h_\ell^r & \end{array} \right)$$

$$= \ (a_0, a_1, b_1, \dots, b_t, c_t, \dots, c_1, d_1, \dots, d_t, e_k, \dots, e_\ell) \quad \in \quad \mathbb{G}^{2+t+2\bar{t}+(\ell-k+1)},$$

where $a'_0, b_1, \dots, b_t, c'_1, \dots, c'_t, d_1, \dots, d_t$ are independent of ID, and y' and z' can be written as, for unknown r',

$$y' \ = \ g_2^\alpha \cdot \left(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3\right)^{r'}, \qquad\qquad z' \ = \ g^{r'} .$$

To generate $d_{\mathsf{ID}}^{(\tau)}$ from $d_{\mathsf{ID}|k-1}^{(\tau)}$, pick a random $r^* \in \mathbb{Z}_p$, and compute

$$y^* \;=\; e_k^{\mathrm{I}_k} \cdot \left(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3\right)^{r^*}, \qquad\qquad z^* \;=\; g^{r^*}.$$

then, output

$$d_{\mathsf{ID}}^{(\tau)} \;=\; \left(\begin{array}{l} a_0 \cdot y^*, \\ a_1 \cdot g^*, \\ b_1, \;\ldots,\; b_t, \\ c_t \cdot y^*, \;\ldots,\; c_1 \cdot y^*, \\ d_1, \;\ldots,\; d_t, \\ e_{k+1} \cdot h_{k+1}^{r^*}, \;\ldots,\; e_\ell \cdot h_\ell^{r^*} \end{array} \right) \;\in\; \mathbb{G}^{2+t+2\bar{t}+(\ell-k)}.$$

If we set $r = r' + r^*$, it is easy to see that this tuple is a correctly distributed private key for identity ID and timestamp τ.

2. $[d_{\mathsf{ID}}^{(\tau')} \to d_{\mathsf{ID}}^{(\tau)}]$. The private key for ID with time index $\tau' < \tau$ can be written as, for unknown exponents $r_1, \ldots, r_t, r'_1, \ldots, r'_t$,

$$\begin{aligned} d_{\mathsf{ID}}^{(\tau')} \;&=\; \left(\begin{array}{ll} y \cdot \left(g_1^{\tau_{t-1}} \cdot f_1\right)^{r_1} \cdot \cdots \cdot \left(g_1^{\tau_1} \cdot f_{t-1}\right)^{r_{t-1}} \cdot \left(g_1^{\tau_0} \cdot f_t\right)^{r_t}, & \\ z, \quad g^{r_1}, \;\ldots,\; g^{r_t}, & \\ y \cdot \left(g_1^{\tau_{t-1}} \cdot f_1\right)^{r_1} \cdot \cdots \cdot \left(g_1 \cdot f_t\right)^{r'_t}, & \text{if } \tau_0 = 0 \\ \quad\vdots & \quad\vdots \\ y \cdot \left(g_1^{\tau_{t-1}} \cdot f_1\right)^{r_1} \cdot \left(g_1 \cdot f_2\right)^{r'_2}, & \text{if } \tau_{t-2} = 0 \\ y \cdot \left(g_1 \cdot f_1\right)^{r'_1}, & \text{if } \tau_{t-1} = 0 \\ g^{r'_1}, \;\ldots,\; g^{r'_t}, & \text{for } g^{r'_i} \text{ where } \tau_{t-i} = 0 \\ e_{k+1}, \;\ldots,\; e_\ell & \end{array} \right) \\ &=\; \left(a_0, a_1, b_1, \ldots, b_t, c_t, \ldots, c_1, d_1, \ldots, d_t, e_{k+1}, \ldots, e_\ell\right) \quad\in\quad \mathbb{G}^{2+t+2\bar{t}+(\ell-k)}, \end{aligned}$$

where y and $e_{k+1}, \ldots, e_\ell$ are independent of the time index τ'.
We need to compute the private key

$$d_{\mathsf{ID}}^{(\tau)} \;=\; (\hat{a}_0, \hat{a}_1, \hat{b}_1, \ldots, \hat{b}_t, \hat{c}_t, \ldots, \hat{c}_1, \hat{d}_1, \ldots, \hat{d}_t, \hat{e}_{k+1}, \ldots, \hat{e}_\ell).$$

The expression of $d_{\mathsf{ID}}^{(\tau)}$ is more clearly described in reference to an instantiation of the HIBE system of Boneh and Boyen [BB04a] where the master key is y (as opposed to explicitly formalizing $d_{\mathsf{ID}}^{(\tau)}$ in terms of $d_{\mathsf{ID}}^{(\tau')}$). Indeed, the $(t+1)$-tuple $[a_0, b_1, \ldots, b_t]$ can be viewed as a private key in the BB system for the level-t identity $\langle \tau'_{t-1} | \ldots | \tau'_0 \rangle$. Similarly, for all $j = t, \ldots, 1$, the $(j+1)$-tuple $[c_j, b_1, \ldots, b_{j-1}, d_j]$ is a private key for $\langle \tau'_{t-1} | \ldots | \tau'_{t-j+1}, 1 \rangle$ when it is defined. It is easy to see that these private keys form a $O(t)$-size "cover set" for the time periods $[\tau', 2^t - 1]$, i.e., for the set of BB-identities $\in \{0,1\}^t$ that represent binary integers $\geq \tau'$. Furthermore, for any $\tau > \tau'$ it is easy to transform this cover set into a $O(t)$-size cover set for the interval $[\tau, 2^t - 1]$ using no more than $2t$ invocations of the (single step) Boneh-Boyen HIBE key derivation procedure.

The abstract transformation from one cover set the other is elementary and is omitted. As for the single-step BB key derivation that implements this transformation, we briefly describe it using the following case as an example. Suppose that the binary expansions of τ' and τ agree on their $j-1$ most significant bits, but differ on the j-th one (then necessarily $\tau'_{t-j} = 0$ and $\tau_{t-j} = 1$). Our cover set for $[\tau', 2^t - 1]$ must contain the BB private key $[c_j, b_1, \ldots, b_{j-1}, d_j]$ corresponding to the BB identity $\langle \tau'_{t-1} | \ldots | \tau'_{t-j+1}, 1 \rangle$. We need to generate the private keys of $\langle \tau'_{t-1} | \ldots | \tau'_{t-j+1}, 1, \beta \rangle$ where $\beta = 0, 1$ (we typically need both). To do so, for each β select a random $r^{(\beta)} \in \mathbb{Z}_p$ and compute the BB private key components:

$$\big(c_j \cdot (g_1^{\beta} \cdot f_{j+1})^{r^*},\ \ b_1,\ \ldots,\ b_{j-1},\ d_j,\ \ g^{r^*}\big).$$

For $\beta = 1$, we get the vector $[\hat{c}_{j+1}, b_1, \ldots, b_{j-1}, \hat{b}_j, \hat{d}_{j+1}]$ whose components can directly be inserted in the appropriate slots in the $d_{\mathsf{ID}}^{(\tau)}$; we also need to insert the value $\hat{d}_{j+1} = g^{r^{(\beta)}}$ in $d_{\mathsf{ID}}^{(\tau)}$. For $\beta = 0$, we only need to insert the value $\hat{b}_{j+1} = g^{r^{(\beta)}}$ in $d_{\mathsf{ID}}^{(\tau)}$; the other components of the private key vector are not directly stored in $d_{\mathsf{ID}}^{(\tau)}$, but are used to obtain the remaining elements of the set cover in a hierarchical manner.

$\mathtt{Encrypt}(params, \mathsf{ID}, M)$**:** To encrypt a message $M \in \mathbb{G}_1$ under the public key $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k) \in \mathbb{Z}_p^k$ with timestamp $\tau = \langle \tau_{t-1} | \cdots | \tau_0 \rangle < 2^t$, pick a random $s \in \mathbb{Z}_p$ and output

$$\begin{aligned} \mathsf{CT} &= \Big(e(g_1, g_2)^s \cdot M,\quad g^s,\ \ \big(h_1^{\mathrm{I}_1} \cdots h_k^{\mathrm{I}_k} \cdot g_3\big)^s,\quad \big(g_1^{\tau_{t-1}} \cdot f_1\big)^s,\ \ \ldots,\ \ \big(g_1^{\tau_0} \cdot f_t\big)^s\ \Big) \\ &\in\ \mathbb{G}_1 \times \mathbb{G}^{2+t}. \end{aligned}$$

$\mathtt{Decrypt}(d_{\mathsf{ID}}, \mathsf{CT})$**:** To decrypt a ciphertext $\mathsf{CT} = (A, B, C, D_1, \ldots, D_t)$ for identity $\mathsf{ID} = (\mathrm{I}_1, \ldots, \mathrm{I}_k)$ with timestamp τ, using a private key $d_{\mathsf{ID}} = (a_0, a_1, b_1, \ldots, b_t, c_t, \ldots, c_1, d_1, \ldots, d_t, e_{k+1}, \ldots, e_\ell)$ for matching ID and τ, output

$$A \cdot e(a_1, C) \cdot \prod_{i=1}^{t} e(d_i, D_i) \Big/ e(B, a_0) = M.$$

The private key components $b_1, \ldots, b_\ell$, $c_t, \ldots, c_1$, and $e_{k+1}, \ldots, e_\ell$ are not used for decryption.

C.2 Security and Complexity

We now mention a few key properties of the linear fs-HIBE scheme.

Security. The security of the fs-HIBE scheme follows immediately from that of the main HIBE presented earlier in this paper (used for the ID hierarchy in the fs-HIBE) and that of the Boneh-Boyen HIBE (used for the temporal set cover construction). We mention that the "reuse" of the BB exponents $r_1, \ldots, r_{t-1}$ in multiple components of $d_{\mathsf{ID}}^{(\tau)}$ is a feature of the subkey generation in the BB system, which is exploited here to reduce the overall size of the full fs-HIBE key; without it, the private key size would incur an extra $\log T$ multiplicative factor.

The proof of security is essentially based on the orthogonality of the two HIBE subsystems in the fs-HIBE construction, from which a reduction to the security of either system is easily obtained. For example, private key derivation in the time coordinate is completely oblivious to the ID-dependent blinding: indeed, as far as the BB subsystem is concerned, the ID-dependent blinding is part of the unknown master secret and left untouched by sub-level key derivation. The same can be said of the timestamp-dependent blinding factors implemented by the BB subsystem, which are unaffected by the key derivations operations in the ID hierarchy. The only place where the two subsystems meet is at decryption, which requires the blinding coefficients from both subsystems to be lifted in order to proceed.

Complexity. For a maximum of T time slices and ℓ levels in the hierarchy, all functions in the fs-HIBE above have time complexity $O(\ell + t) = O(\ell + \log T)$. Similarly, the ciphertexts, the private keys, and the public parameters all have linear size $O(\ell + \log T)$.

C.3 "Logarithmic" Forward Secure Public Key Encryption

We note that the above linear fs-HIBE scheme immediately provides a linear (in t) or logarithmic (in $T = 2^t$) forward secure public key encryption scheme, or fs-PKE. We derive our logarithmic fs-PKE by dropping the ID hierarchy in the above fs-HIBE. In other words, all ciphertexts and private keys are issued to the root of the ID hierarchy. It follows that private keys and ciphertexts in the resulting scheme have complexity $O(\log T)$. Notice that the resulting fs-PKE is entirely based on the Boneh-Boyen IBE scheme from [BB04a], and thus depends only on the decisional BDH assumption.

A difference with the fs-PKE proposed by Canetti et al. [CHK03] is that we achieve logarithmic complexity without resorting to updateable public storage. Recall that in the CHK construction [CHK03], private keys are natively of size $O(\log^2 T)$, and can be compressed down to $O(\log T)$ size at the cost of maintaining polylogarithmic $O(\log^2 T)$ extra information in updateable public storage. The present approach does not require any such public storage.