Hierarchical ID-Based Cryptography

Craig Gentry[1] and Alice Silverberg[2*]

[1] DoCoMo USA Labs
San Jose, CA, USA
cgentry@docomolabs-usa.com

[2] Department of Mathematics
Ohio State University
Columbus, OH, USA
silver@math.ohio-state.edu

Abstract. We present hierarchical identity-based encryption schemes and signature schemes that have total collusion resistance on an arbitrary number of levels and that have chosen ciphertext security in the random oracle model assuming the difficulty of the Bilinear Diffie-Hellman problem.

Keywords: identity-based cryptography, hierarchical identity-based cryptography, elliptic curves, pairings

1 Introduction

1.1 Identity-Based Encryption

This paper answers in the affirmative the question of whether ID-based encryption can be made hierarchical while remaining secure and efficient.

In traditional public key encryption, Bob's public key is a random string unrelated to his identity. When Alice wants to send a message to Bob, she must first obtain Bob's authenticated public key. Typical solutions to this problem involve public key directories. The main idea in identity-based encryption is to eliminate the public key distribution problem by making Bob's public key derivable from some known aspect of his identity, such as his email address. When Alice wants to send a message to Bob, she merely derives Bob's public key directly from his identifying information. Public key directories are unnecessary.

Shamir [15] proposed the idea of identity-based cryptography in 1984, and described an identity-based signature scheme in the same article. However, practical identity-based encryption (IBE) schemes were not found until recently with the work of Boneh and Franklin [3, 4] and Cocks [6] in 2001. Cocks's scheme is based on the "Quadratic Residuosity Problem," and although encryption and decryption are reasonably fast (about the speed of RSA), there is significant message expansion, i.e., the bit-length of the ciphertext is many times the bit-length of the plaintext. The Boneh-Franklin scheme bases its security on the Bilinear Diffie-Hellman Problem, and is quite fast and efficient when using Weil or Tate pairings on supersingular elliptic curves or abelian varieties.

We must note that ID-based encryption has some disadvantages. Bob receives his private key from a third party called a Private Key Generator (PKG) that computes his private key as a function of its master secret and Bob's identity. This requires Bob to authenticate himself to the PKG (in the same way he would authenticate himself to a CA), and requires a secure channel through which the PKG may send Bob his private key. Bob's PKG must publish parameters that embed its master secret, and Alice must obtain these parameters before sending an encrypted

* Silverberg would like to thank DoCoMo USA Labs for support and kind hospitality during her visit there.

message to Bob. Another disadvantage is that the PKG knows Bob's private key, i.e., key escrow is inherent in ID-based systems. Clearly, escrow is a serious problem for some applications.

However, the advantages of identity-based encryption are compelling. The problem of obtaining authentic public keys has been replaced by the problem of obtaining authentic public parameters of PKGs, but the latter should be less burdensome since there will be substantially fewer PKGs than total users. For example, if everyone uses a single PKG, then everyone in the system can communicate securely without ever having to perform online lookup of public keys or public parameters.

1.2 Motivation for Hierarchical ID-Based Encryption (HIDE)

Although having a single PKG would completely eliminate online lookup, it is undesirable for a large network because the PKG has a burdensome job. Not only is private key generation computationally expensive, but also the PKG must verify proofs of identity and must establish secure channels to transmit private keys. Hierarchical ID-based encryption (HIDE) allows a root PKG to distribute the workload by delegating private key generation and identity authentication to lower-level PKGs. In a HIDE scheme, a root PKG need only generate private keys for domain-level PKGs, who in turn generate private keys for users in their domains in the next level. Authentication and private key transmission can be done locally. To encrypt a message to Bob, Alice need only obtain the public parameters of Bob's *root* PKG (and Bob's identifying information); there are no "lower-level parameters." Another advantage of HIDE schemes is damage control: disclosure of a domain PKG's secret does not compromise the secrets of higher-level PKGs. The schemes of Cocks and Boneh-Franklin do not have these properties.

A hierarchical ID-based key sharing scheme with partial collusion-resistance is given in [8, 9]. Horwitz and Lynn [10] introduced hierarchical identity-based encryption, and proposed a 2-level HIDE scheme with total collusion-resistance at the first level and with partial collusion-resistance at the second level, i.e., (a threshold number of) users can collude to obtain the secret of their domain PKG (and thereafter masquerade as the domain PKG). This scheme may be practical for applications where collusion below the first level is not a concern. Finding a secure and practical hierarchical identity-based encryption scheme was, prior to this paper, an important open question.

1.3 Our Results

The scheme in this paper extends the Boneh-Franklin IBE scheme in a natural way. It is a practical, fully scalable, HIDE scheme with total collusion resistance and chosen ciphertext security in the random oracle model, regardless of the number of levels in the hierarchy, assuming the difficulty of the same Bilinear Diffie-Hellman (BDH) problem given in [4] (see Section 2 below). The scheme is quite efficient — the bit-length of the ciphertext and the complexity of decryption grow only linearly with the level of the message recipient.[1] For example, if Bob is at level 1 (just below the root PKG) and Carol is at level 10, Alice's ciphertext to Carol will be about 10 times as long as Alice's ciphertext to Bob, and Carol will take about 10 times as long as Bob to decrypt the message from Alice. At the top level, our HIDE scheme is as fast and efficient as Boneh-Franklin. We show how the scheme can be modified to reduce ciphertext expansion.

The intuitively surprising aspect of this scheme is that, even though lower-level PKGs generate additional random information, this does not necessitate adding public parameters below

[1] Contrast this with [10], where the complexity of encryption grows linearly with the *security* against collusion of a domain PKG's secret. Our scheme has total collusion resistance assuming the difficulty of BDH.

the root level. Also, the random information generated by a lower-level PKG does not adversely affect the ability of users not under the lower-level PKG to send encrypted communications to users under the lower-level PKG.

A hierarchical ID-based signature (HIDS) scheme follows naturally from our HIDE scheme (see Section 4). We also introduce the concept of dual-ID-based encryption (where the ciphertext is a function of both the encrypter and decrypter's identities) and show how this concept, in the context of hierarchical ID-based encryption, allows the length of the ciphertext to be minimized and permits the creation of "escrow shelters" that limit the scope of key escrow.

The rest of the paper is organized as follows. Definitions and background information are given in Section 2. Our hierarchical ID-based encryption scheme is presented in Section 3. An associated hierarchical ID-based signature scheme is given in Section 4. Section 5 gives modifications to minimize the ciphertext expansion. Section 6 discusses how to the restrict the scope of key escrow. Section 7 states results on security, while security proofs, along with variants on the basic schemes, are given in the appendices. Additional extensions and variations are given in Section 8.

2 Definitions

In this section, we give some definitions similar to those given in [3, 4, 10].

ID-tuple: A user has a position in the hierarchy, defined by its tuple of IDs: $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$. The user's ancestors in the hierarchy tree are the root PKG and the users / lower-level PKGs whose ID-tuples are $\{(\mathrm{ID}_1, \ldots, \mathrm{ID}_i) : 1 \leq i < t\}$.

Hierarchical Identity-Based Encryption (HIDE): a HIDE scheme is specified by five randomized algorithms: Root Setup, Lower-level Setup, Extraction, Encryption, and Decryption:

Root Setup: The root PKG takes a security parameter K and returns $params$ (system parameters) and a root secret. The system parameters include a description of the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$. The system parameters will be publicly available, while only the root PKG will know the root secret.

Lower-level Setup: Lower-level users must obtain the system parameters of the root PKG. In HIDE schemes, a lower-level user is not permitted to have any "lower-level parameters" of its own. However, this constraint does not necessarily preclude a lower-level PKG from generating its own lower-level secret, which it may use in issuing private keys to its children. In fact, in our HIDE scheme, a lower-level PKG may generate a lower-level secret, or it may generate random one-time secrets for each Extraction.

Extraction: A PKG (whether the root one or a lower-level one) with ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ may compute a private key for any of its children (e.g., with ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t, \mathrm{ID}_{t+1})$) by using the system parameters and its private key (and any other secret information).

Encryption: A sender inputs $params$, $M \in \mathcal{M}$ and the ID-tuple of the intended message recipient, and computes a ciphertext $C \in \mathcal{C}$.

Decryption: A user inputs $params$, $C \in \mathcal{C}$, and its private key d, and returns the message $M \in \mathcal{M}$.

Encryption and decryption must satisfy the standard consistency constraint, namely when d is the private key generated by the Extraction algorithm for ID-tuple, then:

$$\forall M \in \mathcal{M} : \mathrm{Decryption}(params, d, C) = M \text{ where } C = \mathrm{Encryption}(params, \text{ID-tuple}, M) \ .$$

Hierarchical ID-based Signature (HIDS): a HIDS scheme is specified by five randomized algorithms: Root Setup, Lower-level Setup, Extraction, Signing, and Verification. For Root Setup, the system parameters are supplemented to include a description of the signature space

$\mathcal{S}$. Lower-level Setup and Extraction are as above.
Signing: A signer inputs *params*, its private key d, and $M \in \mathcal{M}$ and outputs a signature $S \in \mathcal{S}$.
Verification: A user inputs *params*, the ID-tuple of the signer, $M \in \mathcal{M}$, and $S \in \mathcal{S}$ and outputs "valid" or "invalid."
Signing and verification must also satisfy a consistency constraint, namely when d is the private key generated by the Extraction algorithm for ID-tuple, then:

$$\forall M \in \mathcal{M} : \text{Verification}(params, \text{ID-tuple}, M, S) = \text{“valid” where } S = \text{Signing}(params, d, M) .$$

The security of our HIDE scheme is based on the difficulty of the Bilinear Diffie-Hellman (BDH) Problem. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of some large prime order q. We write $\mathbb{G}_1$ additively and $\mathbb{G}_2$ multiplicatively. Our HIDE scheme makes use of a "bilinear" pairing.
Admissible pairings: We will call $\hat{e}$ an *admissible pairing* if $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a map with the following properties:

1. Bilinear: $\hat{e}(aQ, bR) = \hat{e}(Q, R)^{ab}$ for all $Q, R \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}$.
2. Non-degenerate: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$.
3. Computable: There is an efficient algorithm to compute $\hat{e}(Q, R)$ for any $Q, R \in \mathbb{G}_1$.

We will also need the mapping $\hat{e}$ to be symmetric, i.e., $\hat{e}(Q, R) = \hat{e}(R, Q)$ for all $Q, R \in \mathbb{G}_1$, but this follows immediately from the bilinearity and the fact that $\mathbb{G}_1$ is a cyclic group. We note that the Weil and Tate pairings associated with supersingular elliptic curves or abelian varieties can be modified to create such bilinear maps, as in [11, 3, 5]; see also [12, 2].
BDH Parameter Generator: As in [3], we say that a randomized algorithm $\mathcal{IG}$ is a BDH parameter generator if $\mathcal{IG}$ takes a security parameter $K > 0$, runs in time polynomial in K, and outputs the description of two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same prime order q and the description of an admissible pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.
Bilinear Diffie-Hellman (BDH) Problem: Given a randomly chosen $P \in \mathbb{G}_1$, as well as aP, bP, and cP (for unknown randomly chosen $a, b, c \in \mathbb{Z}/q\mathbb{Z}$), compute $\hat{e}(P, P)^{abc}$.

For the BDH problem to be hard, $\mathbb{G}_1$ and $\mathbb{G}_2$ must be chosen so that there is no known algorithm for efficiently solving the Diffie-Hellman problem in either $\mathbb{G}_1$ or $\mathbb{G}_2$. Note that if the BDH problem is hard for a pairing $\hat{e}$, then it follows that $\hat{e}$ is non-degenerate.
Bilinear Diffie-Hellman Assumption: As in [3], if $\mathcal{IG}$ is a BDH parameter generator, the advantage $Adv_{\mathcal{IG}}(\mathcal{B})$ that an algorithm $\mathcal{B}$ has in solving the BDH problem is defined to be the probability that the algorithm $\mathcal{B}$ outputs $\hat{e}(P, P)^{abc}$ when the inputs to the algorithm are $\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP, cP$ where $(G_1, \mathbb{G}_2, \hat{e})$ is the output of $\mathcal{IG}$ for sufficiently large security parameter K, P is a random generator of $\mathbb{G}_1$, and a, b, c are random elements of $\mathbb{Z}/q\mathbb{Z}$. The Bilinear Diffie-Hellman assumption is that $Adv_{\mathcal{IG}}(\mathcal{B})$ is negligible for all efficient algorithms $\mathcal{B}$.

3 Hierarchical ID-Based Encryption Schemes

We describe our scheme in a format similar to that used in [4]. We begin by describing a basic scheme, and then extend it to a full scheme that is secure against adaptive chosen ciphertext attack in the random oracle model, assuming the difficulty of the BDH problem.

We may sometimes refer to elements of $\mathbb{G}_1$ as "points," which may suggest that $\hat{e}$ is a modified Weil or Tate pairing, but we note again that any admissible pairing $\hat{e}$ will work.

3.1 BasicHIDE

Let Level_i be the set of entities at level i, where $\text{Level}_0 = \{\text{Root PKG}\}$. Let K be the security parameter given to the setup algorithm, and let $\mathcal{IG}$ be a BDH parameter generator.

Root Setup: The root PKG:

1. runs $\mathcal{IG}$ on input K to generate groups $\mathbb{G}_1, \mathbb{G}_2$ of some prime order q and an admissible pairing $\hat{e}\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$;
2. chooses an arbitrary generator $P_0 \in \mathbb{G}_1$;
3. picks a random $s_0 \in \mathbb{Z}/q\mathbb{Z}$ and sets $Q_0 = s_0 P_0$;
4. chooses cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \mathbb{G}_2 \to \{0,1\}^n$ for some n. The security analysis will treat H_1 and H_2 as random oracles.

The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{G}_1^t \times \{0,1\}^n$ where t is the level of the recipient. The system parameters are $params = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2)$. The root PKG's secret is $s_0 \in \mathbb{Z}/q\mathbb{Z}$.

Lower-level Setup: Entity $\mathrm{E}_t \in \mathrm{Level}_t$ picks a random $s_t \in \mathbb{Z}/q\mathbb{Z}$, which it keeps secret.

Extraction: Let E_t be an entity in Level_t with ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$, where $(\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$ for $1 \le i \le t$ is the ID-tuple of E_t's ancestor at Level_i. Set S_0 to be the identity element of $\mathbb{G}_1$. Then E_t's parent:

1. computes $P_t = H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_t) \in \mathbb{G}_1$;
2. sets E_t's secret point S_t to be $S_{t-1} + s_{t-1} P_t = \sum_{i=1}^{t} s_{i-1} P_i$;
3. also gives E_t the values of $Q_i = s_i P_0$ for $1 \le i \le t-1$.

Encryption: To encrypt $M \in \mathcal{M}$ with the ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$, do the following:

1. Compute $P_i = H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_i) \in \mathbb{G}_1$ for $1 \le i \le t$.
2. Choose a random $r \in \mathbb{Z}/q\mathbb{Z}$.
3. Set the ciphertext to be:

$$C = [rP_0, rP_2, \ldots, rP_t, M \oplus H_2(g^r)] \text{ where } g = \hat{e}(Q_0, P_1) \in \mathbb{G}_2.$$

Decryption: Let $C = [U_0, U_2, \ldots, U_t, V] \in \mathcal{C}$ be the ciphertext encrypted using the ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$. To decrypt C, E_t computes:

$$V \oplus H_2\left(\frac{\hat{e}(U_0, S_t)}{\prod_{i=2}^{t} \hat{e}(Q_{i-1}, U_i)}\right) = M.$$

This concludes the description of our BasicHIDE scheme.

Remark 1. Each lower-level PKG — say, in Level_t — has a secret $s_t \in \mathbb{Z}/q\mathbb{Z}$, just like the root PKG. A lower-level PKG uses this secret to generate a secret point for each of its children, just as the root PKG does. An interesting fact, however, is that lower-level PKGs need not always use the same s_t for each private key extraction. Rather, s_t could be generated randomly for each of the PKG's children.

Remark 2. H_1 can be chosen to be an iterated hash function so that, for example, P_i may be computed as $H_1(P_{i-1}, \mathrm{ID}_i)$ rather than $H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$.

Remark 3. In what follows, we may refer to S_t as E_t's *private point*, and to $\{Q_i : 1 \le i < t\}$ as E_t's *Q-values*. We say that S_t' and $\{Q_i' : 1 \le i < t\}$ form a *valid private key* for the point-tuple $(P_1, \ldots, P_t)$ if $S_t' = s_0 P_1 + \sum_{i=1}^{t-1} s_i' P_{i+1}$ and $Q_i' = s_i' P_0$ for some $(s_1', \ldots, s_{t-1}') \in (\mathbb{Z}/q\mathbb{Z})^{t-1}$.

3.2 FullHIDE: HIDE with Chosen Ciphertext Security

In [4], Fujisaki-Okamoto padding [7] is used to convert a basic IBE scheme to an IBE scheme that is chosen ciphertext secure in the random oracle model. In the same way, BasicHIDE can be converted to FullHIDE, a HIDE scheme that is chosen ciphertext secure in the random oracle model. Next we describe the scheme FullHIDE.

Setup: As in the BasicHIDE scheme, but in addition choose hash functions $H_3 : \{0,1\}^n \times \{0,1\}^n \to \mathbb{Z}/q\mathbb{Z}$ and $H_4 : \{0,1\}^n \to \{0,1\}^n$.

Extraction: As in the BasicHIDE scheme.

Encryption: To encrypt $M \in \mathcal{M}$ with the ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$, do the following:

1. compute $P_i = H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_i) \in \mathbb{G}_1$ for $1 \le i \le t$,
2. choose a random $\sigma \in \{0,1\}^n$,
3. set $r = H_3(\sigma, M)$, and
4. set the ciphertext to be:

$$C = [rP_0, rP_2, \ldots, rP_t, \sigma \oplus H_2(g^r), M \oplus H_4(\sigma)]$$

   where $g = \hat{e}(Q_0, P_1) \in \mathbb{G}_2$ as before.

Decryption: Let $C = [U_0, U_2, \ldots, U_t, V, W] \in \mathcal{C}$ be the ciphertext encrypted using the ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$. If $(U_0, U_2, \ldots, U_t) \notin \mathbb{G}_1^t$, reject the ciphertext. To decrypt C, E_t does the following:

1. computes

$$V \oplus H_2\left(\frac{\hat{e}(U_0, S_t)}{\prod_{i=2}^{t} \hat{e}(Q_{i-1}, U_i)}\right) = \sigma,$$

2. computes $W \oplus H_4(\sigma) = M$,
3. sets $r = H_3(\sigma, M)$ and tests that $[U_0, U_2, \ldots, U_t, V]$ is a BasicHIDE encryption of M using r and $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$. If not, it rejects the ciphertext.
4. outputs M as the decryption of C.

Note that M is encrypted as $W = M \oplus H_4(\sigma)$. This can be replaced by $W = E_{H_4(\sigma)}(M)$ where E is a semantically secure symmetric encryption scheme (see [7] and Section 4.2 of [4]).

4 Hierarchical ID-based Signature (HIDS) Schemes

ID-based encryption, whether hierarchical or not, has a clear advantage over PKI; it does not require online public key lookup. On the other hand, it is not so clear that ID-based signatures have an advantage over traditional signature schemes using PKI. Indeed, any public-key signature scheme may be transformed into an ID-based (hierarchical) signature scheme by using (a hierarchy of) certificates, since certificates "bind" an identity to a public key.

The previous comments notwithstanding, we present a Hierarchical ID-based Signature (HIDS) scheme based on the difficulty of solving the Diffie-Hellman problem in the group $\mathbb{G}_1$. When viewed in isolation, this HIDS scheme is not especially useful for the reasons stated above (though it may be more efficient). However, as will be explained later, the HIDS scheme becomes quite useful when viewed in combination with the HIDE scheme as a complete package.

4.1 A HIDS Scheme

As noted by Moni Naor (see Section 6 of [4]), an IBE scheme can be immediately converted into a public key signature scheme as follows: the signer's private key is the master key in the IBE scheme. The signer's signature on M is the IBE decryption key d corresponding to the "public key" $H_1(\mathrm{ID}) = H_1(M)$. The verifier checks the signature by choosing a random message M', encrypting M' with $H_1(M)$, and trying to decrypt the resulting ciphertext with d. If the ciphertext decrypts correctly, the signature is considered valid.

This observation can be extended to a hierarchical context: a HIDE scheme can be immediately converted to a HIDS scheme. Suppose the signer has ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$. To sign M, the signer computes a private key d for the ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t, M)$, and sends d to the verifier. As before, the verifier checks the signature by choosing a random message M', encrypting M' with the "public key" $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t, M)$, and trying to decrypt the resulting ciphertext with d. The security of this HIDS scheme follows immediately from the security of our HIDE scheme, since forging a signer's signature is equivalent to recovering the private key of one of the signer's children.

An obvious pitfall in the HIDS scheme just described is that an attacker might try to get the signer to sign $M = \mathrm{ID}_{t+1}$ where ID_{t+1} represents an actual identity. In this case, the signer's signature will actually be a private key, which thereafter may be used to decrypt messages and forge signatures. The easy solution to this problem is to use some expedient — such as a bit prefix — that distinguishes between signing and private key extraction.

Below we describe our HIDS scheme in more detail. The security of the HIDS scheme is based on the difficulty of solving the Diffie-Hellman problem in the group $\mathbb{G}_1$ (as opposed to HIDE, which requires the BDH problem to be difficult, and therefore requires the Diffie-Hellman problem in $\mathbb{G}_2$ to be difficult).

Let Level_i be the set of entities at level i, where $\mathrm{Level}_0 = \{\text{Root PKG}\}$. Let K be the security parameter given to the setup algorithm, and let $\mathcal{IG}$ be a BDH parameter generator.

Root Setup: The root PKG:

1. runs $\mathcal{IG}$ on input K to generate groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order q and an admissible pairing $\hat{e}\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$;
2. chooses an arbitrary generator $P_0 \in \mathbb{G}_1$;
3. picks a random $s_0 \in \mathbb{Z}/q\mathbb{Z}$ and sets $Q_0 = s_0 P_0$;
4. chooses cryptographic hash functions $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_3 : \{0,1\}^* \to \mathbb{G}_1$. The security analysis will treat H_1 and H_3 as random oracles.

The signature space is $\mathcal{S} = \mathbb{G}_1^{t+1}$ where t is the level of the signer. The system parameters are $params = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_3)$. The root PKG's secret is $s_0 \in \mathbb{Z}/q\mathbb{Z}$.

Lower-level Setup: As in BasicHIDE.

Extraction: As in BasicHIDE.

Signing: To sign M with ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ (using the secret point $S_t = \sum_{i=1}^{t} s_{i-1} P_i$ and the points $Q_i = s_i P_0$ for $1 \le i \le t$), do the following:

1. Compute $P_M = H_3(\mathrm{ID}_1, \ldots, \mathrm{ID}_t, M) \in \mathbb{G}_1$. (As suggested above, we might use a bit-prefix or some other method, instead of using a totally different hash function.)
2. Compute $Sig(\text{ID-tuple}, M) = S_t + s_t P_M$.
3. Send $Sig(\text{ID-tuple}, M)$ and $Q_i = s_i P_0$ for $1 \le i \le t$.

Verification: Let $[Sig, Q_1, \ldots, Q_t] \in \mathcal{S}$ be the signature for (ID-tuple, M). The verifier confirms that:

$$\hat{e}(P_0, Sig) = \hat{e}(Q_0, P_1)\hat{e}(Q_t, P_M)\prod_{i=2}^{t}\hat{e}(Q_{i-1}, P_i).$$

5 Shortening the Ciphertext and Signatures

In the HIDE scheme, the length of the ciphertext is proportional to the depth of the recipient in the hierarchy. Similarly, in the hierarchical ID-based signature scheme, the length of the signature is proportional to the depth of the signer in the hierarchy, unless the verifier already has the signer's Q-values. This section discusses ways in which this ciphertext expansion problem may be avoided.

5.1 Dual-HIDE: Dual-Identity-Based Encryption

In 2000, Sakai, Ohgishi and Kasahara [14] presented a "key sharing scheme" based on the Weil pairing. The idea was quite simple: suppose a PKG has a master secret s, and it issues private keys to users of the form sP_y, where $P_y = H_1(\mathrm{ID}_y)$ and ID_y is the ID of user y (as in Boneh-Franklin). Then users y and z have a shared secret that only they (and the PKG) may compute, namely, $\hat{e}(sP_y, P_z) = \hat{e}(P_y, P_z)^s = \hat{e}(P_y, sP_z)$. They may use this shared secret to encrypt their communications. Notice that this key sharing scheme does not require any interaction between the parties. We can view Sakai, Ohgishi and Kasahara's discovery as a type of "dual-identity-based encryption," where the word "dual" indicates that the identities of both the sender and the recipient (rather than just the recipient) are required as input into the encryption and decryption algorithms. The main practical difference between this scheme and the Boneh-Franklin IBE scheme is that the sender must obtain its *private key* from the PKG before sending encrypted communications, as opposed to merely obtaining the *public parameters* of the PKG.

In the non-hierarchical context, Dual-IBE does not appear to have substantial advantages over IBE. In the hierarchical context, however, Dual-HIDE may be more efficient than HIDE if the sender and recipient are close to each other in the hierarchy tree. Suppose two users, y and z, have the ID-tuples $(\mathrm{ID}_{y1}, \ldots, \mathrm{ID}_{yl}, \ldots, \mathrm{ID}_{ym})$ and $(\mathrm{ID}_{z1}, \ldots, \mathrm{ID}_{zl}, \ldots, \mathrm{ID}_{zn})$, where $(\mathrm{ID}_{y1}, \ldots, \mathrm{ID}_{yl}) = (\mathrm{ID}_{z1}, \ldots, \mathrm{ID}_{zl})$. In other words, user y is in Level_m, user z is in Level_n, and they share a common ancestor in Level_l. User y may use Dual-HIDE to encrypt a message to user z as follows:

Encryption: To encrypt $M \in \mathcal{M}$, user y:

1. Computes $P_{zi} = H_1(\mathrm{ID}_{z1}, \ldots, \mathrm{ID}_{zi}) \in \mathbb{G}_1$ for $l+1 \leq i \leq n$.
2. Chooses a random $r \in \mathbb{Z}/q\mathbb{Z}$.
3. Sets the ciphertext to be:

$$C = [rP_0, rP_{z(l+1)}, \ldots, rP_{zn}, M \oplus H_2(g_{yl}^r)]$$

where

$$g_{yl} = \frac{\hat{e}(P_0, S_y)}{\prod_{i=l+1}^{m}\hat{e}(Q_{y(i-1)}, P_{yi})} = \hat{e}(P_0, S_{yl})\ ,$$

S_y is y's secret point, S_{yl} is the secret point of y's and z's common ancestor at level l, and $Q_{yi} = s_{yi}P_0$ where s_{yi} is the secret number chosen by y's ancestor at level i.

Decryption: Let $C = [U_0, U_{l+1}, \ldots, U_n, V]$ be the ciphertext. To decrypt C, user z computes:

$$V \oplus H_2\left(\frac{\hat{e}(U_0, S_z)}{\prod_{i=l+1}^{n} \hat{e}(Q_{z(i-1)}, U_i)}\right) = M.$$

Note that if y and z have a common ancestor below the root PKG, then the ciphertext is shorter with Dual-HIDE than with non-dual HIDE. Further, using Dual-HIDE, the encrypter y computes $m - l + 1$ pairings while the decrypter z computes $n - l + 1$ pairings. (Note that $m + n - 2l$ is the "length" of the path between y and z in the hierarchy tree.) In the non-dual HIDE scheme, the encrypter computes one pairing while the decrypter computes n pairings. Thus when $m < 2l - 1$, the total work is less with Dual-HIDE than with non-dual HIDE. The relative computing power of the sender and recipient can also be taken into account. In Appendix B, we show how to decrease the number of pairings that y and z must compute to $m+n-2l+1$ if their common ancestor in Level_l always uses the same s_l rather than generating this number randomly with each private key extraction.

Dual-HIDE also makes domain-specific broadcast encryption possible. Suppose user y wants to encrypt a message to everyone having the same ancestor in Level_l. Everyone in this common ancestor's domain may compute the shared secret $\hat{e}(P_0, S_{yl})$, and so this secret may be used as a shared key of everyone in this domain. Users outside of this domain, other than the parent of the common ancestor, will be unable to compute this pairing. (In Section 6.1, we describe how to exclude even the parent.) Note that Dual-HIDE broadcast is not fully compatible with the HIDS scheme. If Dual-HIDE broadcast and the HIDS scheme use the same parameters, everyone outside the domain who receives a signature from someone in the domain will also be able to compute $\hat{e}(P_0, S_{yl})$.

Fujisaki-Okamoto padding turns Dual-HIDE into FullDual-HIDE, a dual-identity encryption scheme with adaptive chosen ciphertext security.

5.2 Dual-HIDS: Dual-Identity-Based Signatures

Dual hierarchical identity-based signatures (Dual-HIDS) are much easier to explain. If users y and z, as above, have a common ancestor in Level_l, then y only needs to send Q_{yi} for $l + 1 \leq i \leq m$. This makes the length of the signature proportional to $m - l$ rather than m.

5.3 Authenticated Lower-level Root PKGs

Suppose that user y often sends mail to people at a certain university — say, Cryptography State University (CSU) — but that CSU is deep in the hierarchy, and that y is not close to CSU in the hierarchy. How do we solve the ciphertext expansion problem? One solution, of course, is for CSU to set up its own root PKG with its own system parameters, unassociated with the "actual" root PKG. After y obtains CSU's system parameters, its ciphertext to CSU recipients will be shorter. However, we would prefer not to have "rogue" root PKGs.

A better solution is for CSU to set up a root PKG that is "authenticated" by the actual root PKG. For this purpose, the actual root PKG may have an additional parameter, a random message M'. To set up its authenticated root PKG, CSU "signs" M', generating the signature $Sig = S_t + s_t P_{M'}$, where S_t is CSU's private point, and s_t is its lower-level secret. CSU also publishes Q_i for $1 \leq i \leq t$.

Let $(\text{ID}_1, \ldots, \text{ID}_t, \ldots, \text{ID}_v)$ be the ID-tuple of user z at CSU having point-tuple $(P_1, \ldots, P_t, \ldots, P_v)$. Then y may send an encrypted message to z, using the parameters for CSU's authenticated root PKG, as follows:

Encryption: To encrypt $M \in \mathcal{M}$, user y:

1. Computes $P_i = H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_i) \in \mathbb{G}_1$ for $t+1 \leq i \leq v$.
2. Chooses a random $r \in \mathbb{Z}/q\mathbb{Z}$.
3. Sets the ciphertext to be:

$$C = [rP_0, rP_{t+1}, \ldots, rP_v, M \oplus H_2(g_t^r)] \text{ where } g_t = \frac{\hat{e}(P_0, Sig)}{\hat{e}(s_t P_0, P_{M'})} = \hat{e}(P_0, S_t) \ .$$

Decryption: Let $C = [U_0, U_{t+1}, \ldots, U_v, V]$ be the ciphertext. To decrypt C, user z computes:

$$V \oplus H_2\left(\frac{\hat{e}(U_0, S_v)}{\prod_{i=t+1}^{v} \hat{e}(Q_{i-1}, U_i)}\right) = M$$

where S_v is z's private key. The number of pairings computed by the decrypter is $v - t + 1$, one more than its depth below CSU , not its depth below the actual root PKG.

Interestingly, if y obtains *any* signature from CSU , not necessarily on a particular message M', then y may use that signature to shorten its ciphertext in the same way. In effect, y's possession of any signature from CSU allows y to use Dual-HIDE as if y's position in the hierarchy is just below CSU . Thus, y may use CSU 's signature to shorten its ciphertext not only to entities below CSU in the hierarchy, but also to any entity that is *close* to CSU in the hierarchy. In general, one could have an "optimized" HIDE scheme in which the sender stores a list of HIDS signatures that it has obtained, and, upon each encryption, searches through that list (which may be put in lexicographic order) to find the signer that is closest in the hierarchy to the intended message recipient, and then uses that signer's signature, in combination with Dual-HIDE, to minimize the length of the ciphertext.

6 Restricting Key Escrow

In IBE schemes, key escrow is "inherent" because the PKG knows the private key of each user. Even in the hierarchical scheme of Horwitz and Lynn, every ancestor of a given user in the hierarchy knows that user's private key. Although this key escrow property may be useful in some contexts, it is certainly not desirable for all applications.

In our HIDE scheme, since the private point of a user depends on a secret number known only to the parent of that user, no ancestor other than the parent may compute the user's particular private point. However, the user's ancestors can still decrypt the user's mail; they may simply compute a different (but equally effective) private key for the user based on different lower-level Q-values. Using these different Q-values, they may also forge the user's signature. In this section, we discuss how Dual-HIDE and/or key agreement protocols can be used to restrict this key escrow property.

6.1 Restricting Key Escrow using Dual-HIDE

Consider again users y and z from Section 5.1 who have a common ancestor in Level_l. Let's say their common ancestor is Cryptography State University, and suppose that user y uses Dual-HIDE to encrypt its messages to z. As stated above, CSU 's parent knows CSU 's private point. From CSU 's perspective, this may be an undesirable situation. However, CSU can easily change its private point S_l by setting $S_l := S_l + bP_l$ and setting $Q_{l-1} := Q_{l-1} + bP_0$ for some random $b \in \mathbb{Z}/q\mathbb{Z}$. This new private key is just as effective, and is unknown to CSU 's parent. Assuming that CSU uses its new private key to issue private keys to its children, none of CSU 's ancestors will be able to decrypt y's message to z encrypted using Dual-HIDE. More specifically, only ancestors of z that are within CSU 's domain will be able to decrypt.

6.2 Authenticated Key Agreement with no Session Key Escrow

HIDS provides a convenient platform on which key agreement may be authenticated (see also [1] for authenticated three-party (non-ID based) key agreement protocols using pairings). A simple explicit authenticated key agreement protocol is as follows:

1. Alice chooses a random $a \in \mathbb{Z}/q\mathbb{Z}$ and sends aP_0 and $Sign(aP_0)$ to Bob.
2. Bob chooses a random $b \in \mathbb{Z}/q\mathbb{Z}$ and sends bP_0 and $Sign(bP_0)$ to Bob.
3. Alice and Bob verify the received signatures and compute the shared secret: abP_0.

Here, there is no session key escrow. However, there is still an attack scenario: an ancestor of Alice and an ancestor of Bob could collude to mount a man-in-the-middle attack. This attack has an analogue in PKI: CAs could collude in a similar way. Dual-HIDE can be used in combination with key agreement to minimize the possible scope of such collusion among ancestors.

Implicit authentication based on Sakai-Ohgishi-Kasahara key agreement can be done as follows. Alice and Bob first perform a standard (or elliptic curve) Diffie-Hellman exchange, after which Alice thinks the shared Diffie-Hellman value is g_A and Bob thinks it is g_B. Then Alice computes the shared secret as $H(g_A, S_{AB})$ and Bob computes it as $H(g_B, S_{AB})$, where H is a one-way collision-resistant hash function and $S_{AB} = \hat{e}(P_A, P_B)^s = \hat{e}(S_A, P_B) = \hat{e}(S_B, P_A)$, where $P_A = H_1(\mathrm{ID}_A)$ is Alice's public point and $S_A = sP_A$ is her private point, $P_B = H_1(\mathrm{ID}_B)$ is Bob's public point and $S_B = sP_B$ is Bob's private point, and s is their PKG's master secret. Unless the man-in-the-middle is the PKG, it will not be able to compute Alice's or Bob's version of the shared secret, since it does not know S_{AB}. However, it can prevent Alice and Bob from computing the same shared secret. Alice and Bob will not know that their key agreement protocol has been disrupted until one sends an undecipherable message to the other. A passive PKG will not know Alice's and Bob's shared Diffie-Hellman value, and is therefore unable to compute the session key.

7 Security

The security of BasicHIDE and Dual-HIDE (see Appendix A for security proofs and definitions of terminology) is based on the difficulty of the BDH problem, as stated in the following theorems (which are analogous to Theorem 4.1 in [4]):

Theorem 1. *Suppose there is an NHID-OWE adversary $\mathcal{A}$ that has advantage ϵ against the BasicHIDE or Dual-HIDE scheme for some ID-tuple and that makes $q_{H_2} > 0$ hash queries to the hash function H_2 and a finite number of private key extraction queries. If the hash functions H_1, H_2 are random oracles, then there is an algorithm $\mathcal{B}$ that solves the BDH in groups generated by $\mathcal{IG}$ with advantage at least $(\epsilon - \frac{1}{2^n})/q_{H_2}$ and running time $O(time(\mathcal{A}))$.*

Theorem 2. *Suppose there is an HID-OWE adversary $\mathcal{A}$ that makes at most $q_{H_2} > 0$ hash queries to the hash function H_2 and at most $q_E > 0$ private key extraction queries and has advantage ϵ_t of successfully targeting a BasicHIDE or Dual-HIDE node in Level$_t$. If the hash functions H_1, H_2 are random oracles, then there is an algorithm $\mathcal{B}$ that solves the BDH in groups generated by $\mathcal{IG}$ with advantage at least $(\epsilon_t (\frac{t}{e(q_E+t)})^t - \frac{1}{2^n}) q_{H_2}^{-1}$ and running time $O(time(\mathcal{A}))$.*

With Fujisaki-Okamoto padding, these schemes can be made chosen ciphertext secure if BDH is hard in the groups generated by $\mathcal{IG}$. The proof follows from Theorems 1 and 2 analogously to the way that Theorem 4.4 of [4] follows from Lemma 4.3 of [4]. Further, the security of the HIDS scheme depends only on the difficulty of the Diffie-Hellman problem in the group $\mathbb{G}_1$, and not on BDH.

8 Extensions and Observations

Improving efficiency of encryption: Levels 0 and 1 can be merged into a single (combined levels 0 and 1) root PKG. In that case, $g = \hat{e}(Q_0, P_1)$ is included in the system parameters. This saves encrypters the task of computing the value of this pairing. However, decrypters must compute an extra pairing (as a result of being one level lower down the tree).

Distributed PKGs: As in Section 6 of [4], the secrets s_i and private keys can be distributed using techniques of threshold cryptography to protect the secrets and make the scheme robust against dishonest PKGs.

Concrete Schemes: The same elliptic curves that were used to give IBE schemes in [4], or the elliptic curves that were used to give short signature schemes in [5], or the abelian varieties given in [13], give HIDE and HIDS schemes.

9 Conclusion

We gave hierarchical ID-based encryption (HIDE) schemes that are practical, totally collusion-resistant, and secure against chosen-ciphertext attacks. The message expansion factor and complexity of decryption grow only linearly with the number of levels in the hierarchy. We introduced a related hierarchical ID-based signature (HIDS) scheme that is especially effective when used in combination with HIDE and Dual-HIDE. This also appears to be the first paper related to ID-based cryptography that gives methods for circumventing key escrow.

Acknowledgments: We thank Jonathan Katz, Yiqun Lisa Yin, James Kempf, Anand Desai, and Satomi Okazaki for helpful conversations.

References

1. S. S. Al-Riyami and K. G. Paterson. Authenticated Three Party Key Agreement Protocols from Pairings. Cryptology e-Print Archive, `http://eprint.iacr.org/2002/035/` .
2. P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient Algorithms for Pairing-Based Cryptosystems, to appear in Crypto 2002.
3. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing, in Advances in Cryptology — Crypto 2001, Lecture Notes in Computer Science **2139** (2001), Springer, 213–229.
4. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing, extended version of [3], `http://www.cs.stanford.edu/~dabo/papers/ibe.pdf` .
5. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing, in Advances in Cryptology — Asiacrypt 2001, Lecture Notes in Computer Science **2248** (2001), Springer, 514–532.
6. C. Cocks. An identity based encryption scheme based on quadratic residues, in Cryptography and Coding, Lecture Notes in Computer Science **2260** (2001), Springer, 360–363.
7. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes, in Advances in Cryptology — Crypto '99, Lecture Notes in Computer Science **1666** (1999), Springer, 537–554.
8. G. Hanaoka, T. Nishioka, Y. Zheng, and H. Imai. An efficient hierarchical identity-based key-sharing method resistant against collusion-attacks, in Advances in Cryptology — Asiacrypt 1999, Lecture Notes in Computer Science **1716** (1999), Springer, 348–362.
9. G. Hanaoka, T. Nishioka, Y. Zheng, and H. Imai, A hierarchical non-interactive key-sharing scheme with low memory size and high resistance against collusion attacks, to appear in *The Computer Journal*.
10. J. Horwitz and B. Lynn. Toward Hierarchical Identity-Based Encryption, to appear in Advances in Cryptology — Eurocrypt 2002, Lecture Notes in Computer Science **2332** (2002), Springer, 466–481.
11. A. Joux. A one round protocol for tripartite Diffie-Hellman, in Algorithmic Number Theory (ANTS-IV), Lecture Notes in Computer Science **1838** (2000), Springer, 385–394.
12. V. Miller. Short programs for functions on curves, unpublished manuscript.
13. K. Rubin and A. Silverberg. Supersingular abelian varieties in cryptology, to appear in Crypto 2002.

14. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. SCIC 2000-C20, Okinawa, Japan, January 2000.
15. A. Shamir. Identity-based cryptosystems and signature schemes, in Advances in Cryptology — Crypto '84, Lecture Notes in Computer Science **196** (1984), Springer, 47–53.

A Proofs of Security

A.1 Security Definitions

We first give some definitions that are very similar to those given in [3, 4, 10]. Their similarity should not be surprising because, *at a high level*, the security issues involved in hierarchical ID-based cryptography are substantially identical to those in non-hierarchical ID-based cryptography; we are merely adding new levels.

Chosen-ciphertext security: As Boneh and Franklin noted in the context of (non-hierarchical) ID-based cryptography, the standard definition of chosen-ciphertext security must be strengthened for ID-based systems, since one should assume that an adversary can obtain the private key associated with any identity of its choice (other than the particular identity being attacked). The same applies to hierarchical ID-based cryptography. Thus, we allow an attacker to make "private key extraction queries." Also, as in [4], we allow the adversary to choose the identity on which it wishes to be challenged.

One subtlety is that an adversary may choose its target identity adaptively or nonadaptively. An adversary that chooses its target adaptively will first make hash queries and extraction queries, and then choose its target based on the results of these queries. Such an adversary might not have a particular target in mind when it begins the attack, and its eventual target need not even belong to an existing entity. Rather, this adversary is successful if it is able to hack some identity to which it is not entitled. A nonadaptive adversary, on the other hand, chooses its target independently from results of hash queries and extraction queries. For example, such an adversary might target a personal enemy. The adversary may still make hash and extraction queries, but its target choice is based strictly on the target's identity, not on query results. Obviously, security against an adaptively-chosen-target adversary is the stronger, and therefore preferable, notion of security. However, we will address both types of security, since our security proofs against nonadaptively-chosen-target adversaries are stronger.

We say that a HIDE scheme is semantically secure against adaptive chosen ciphertext and adaptive (resp., nonadaptive) chosen target attack (IND-HID-CCA (resp. IND-NHID-CCA)) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following game. (Note: for IND-NHID-CCA, Phase 1 is omitted.)

Setup: The challenger takes a security parameter K and runs the Root Setup algorithm. It gives the adversary the resulting system parameters *params*. It keeps the root key to itself.

Phase 1: The adversary issues queries $q_1, \ldots, q_m$ where q_i is one of:

1. Public-key query (ID-tuple$_i$): The challenger runs a hash algorithm on ID-tuple$_i$ to obtain the public key H(ID-tuple$_i$) corresponding to ID-tuple$_i$.
2. Extraction query (ID-tuple$_i$): The challenger runs the Extraction algorithm to generate the private key d_i corresponding to ID-tuple$_i$, and sends d_i to the adversary.
3. Decryption query (ID-tuple$_i$,C_i): The challenger runs the Extraction algorithm to generate the private key d_i corresponding to ID-tuple$_i$, runs the Decryption algorithm to decrypt C_i using d_i, and sends the resulting plaintext to the adversary.

These queries may be asked adaptively. Note also that the queried ID-tuple$_i$ may correspond to a position at any level in the hierarchy.

Challenge: Once the adversary decides that Phase 1 is over, it outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ and an ID-tuple on which it wishes to be challenged. The only constraints

are that neither this ID-tuple nor its ancestors appear in any private key extraction query in Phase 1. Again, this ID-tuple may correspond to a position at any level in the hierarchy.
The challenger picks a random bit $b \in \{0,1\}$ and sets $C = \text{Encryption}(params, \text{ID-tuple}, M_b)$. It sends C as a challenge to the adversary.
Phase 2: The adversary issues more queries $q_{m+1}, \ldots, q_n$ where q_i is one of:

1. Public-key query (ID-tuple$_i$): Challenger responds as in Phase 1.
2. Extraction query (ID-tuple$_i$ $\neq$ ID-tuple or ancestor): Challenger responds as in Phase 1.
3. Decryption query ((ID-tuple$_i$,C_i) $\neq$ (ID-tuple or ancestor,C)): Challenger responds as in Phase 1.

Guess: The adversary outputs a guess $b' \in \{0,1\}$. The adversary wins the game if $b = b'$. We define its advantage in attacking the scheme to be $|Pr[b = b'] - \frac{1}{2}|$.

One way identity-based encryption: As in [3], we define one-way encryption (OWE) for a public key encryption scheme as follows. The adversary $\mathcal{A}$ is given a random public key K_{pub} and a ciphertext C that is the encryption of a random message M using K_{pub}, and outputs a guess for the plaintext. The adversary is said to have advantage ϵ against the scheme if ϵ is the probability that $\mathcal{A}$ outputs M. The scheme is said to be a one-way encryption (OWE) scheme if no polynomial time adversary has a non-negligible advantage in attacking the scheme.

We say that a HIDE scheme is one-way (HID-OWE or NHID-OWE, depending on whether the target is chosen adaptively or not) if no polynomial time adversary has a non-negligible advantage against the challenger in the following game. (Phase 1 is omitted for NHID-OWE.)
Setup: The challenger takes a security parameter k and runs the Root Setup algorithm. It gives the adversary the resulting system parameters *params*. It keeps the root key to itself.
Phase 1: The adversary makes public-key and/or extraction queries as in Phase 1 above.
Challenge: Once the adversary decides that Phase 1 is over, it outputs a new ID-tuple on which it wishes to be challenged. The challenger picks a random $M \in \mathcal{M}$ and sets $C = \text{Encryption}(params, \text{ID-tuple}, M)$. It sends C as a challenge to the adversary.
Phase 2: The adversary issues more public-key queries and more extraction queries on identities other than this ID-tuple and its ancestors, and the challenger responds as in Phase 1.
Guess: The adversary outputs a guess $M' \in \mathcal{M}$. The adversary wins the game if $M = M'$. We define the adversary's advantage in attacking the scheme to be $Pr[M = M']$.

Security against Existential Forgery on Adaptively Chosen Messages: An adversary should be unable to forge its target's signature on a message that the target has not signed previously, even after (adaptively) obtaining the target's signature on messages of the adversary's choosing. A HIDS adversary also has the ability to choose its target, and to make public key queries and private key extraction queries on entities other than the target and its ancestors. As with HIDE, the adversary's choice of target may be adaptive or nonadaptive.

A.2 BasicPub

To analyze the security of BasicHIDE, we first define a related public-key encryption scheme called BasicPub. This scheme is exactly the same as the "BasicPub" presented in [4], but we present it again here because our notation is slightly different. We will then use two lemmas — one proving that breaking BasicHIDE is as hard as breaking BasicPub, the other proving that breaking BasicPub is as hard as solving an instance of the BDH problem — to show that the security of BasicHIDE is based on the difficulty of the BDH problem.

BasicPub is a public-key encryption scheme, specified by three algorithms: Key Generation, Encryption, and Decryption:

Key Generation:

1. Run $\mathcal{IG}$ on input K to generate two groups $\mathbb{G}_1, \mathbb{G}_2$ of the same prime order q and a bilinear map $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Choose an arbitrary generator $P_0 \in \mathbb{G}_1$.
2. Pick a random $s_0 \in \mathbb{Z}/q\mathbb{Z}$ and set $Q_0 = s_0 P_0$.
3. Pick a random point $P_1 \in \mathbb{G}_1$.
4. Choose a cryptographic hash function $H_2 : \mathbb{G}_2 \to \{0,1\}^n$.

The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{G}_1 \times \{0,1\}^n$. The public key is $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, P_1, H_2)$. The private key is $S_1 = s_0 P_1$.

Encryption: To encrypt $M \in \mathcal{M}$, choose a random $r \in \mathbb{Z}/q\mathbb{Z}$ and set the ciphertext to be:

$$C = [rP_0, M \oplus H_2(g^r)] \text{ where } g = \hat{e}(Q_0, P_1) \in \mathbb{G}_2.$$

Decryption: Let $C = [U, V] \in \mathcal{C}$ be the ciphertext. To decrypt C, compute:

$$V \oplus H_2(\hat{e}(U, S_1)) = M.$$

A.3 HIDE: Targeting a Specific User

Lemma 1. *Suppose that $\mathcal{A}$ is an NHID-OWE adversary that makes a finite number of private key extraction queries and has advantage ϵ against BasicHIDE for some ID-tuple, and suppose that the hash function H_1 is a random oracle. Then there is an OWE adversary $\mathcal{B}$ with advantage at least ϵ against BasicPub and running time $O(time(\mathcal{A}))$.*

Proof: We show how to construct an OWE adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage ϵ against BasicPub. The game between the challenger and the adversary $\mathcal{B}$ starts with the challenger first generating a random public key by running the key generation algorithm of BasicPub. The result is a public key $K_{pub} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, P_1, H_2)$, with $Q_0 = s_0 P_0$, and a private key $S_1 = s_0 P_1$. The challenger then picks a random plaintext $M \in \mathcal{M}$ and encrypts M using the encryption algorithm of BasicPub. It gives K_{pub} and the resulting ciphertext $C = [U, V]$ to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ is supposed to output a guess for M. Let ID-tuple$_0 = (\mathrm{ID}_{01}, \ldots, \mathrm{ID}_{0t_0})$ be an ID-tuple for which $\mathcal{A}$ has advantage ϵ against BasicHIDE. Then $\mathcal{A}$ and $\mathcal{B}$ agree to use ID-tuple$_0$, and they interact as follows:
Setup: $\mathcal{B}$ gives $\mathcal{A}$ the BasicHIDE system parameters $params = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2)$. Here, H_1 is a random oracle controlled by $\mathcal{B}$.
H_1-queries: At any time, algorithm $\mathcal{A}$ can query the random oracle H_1. This oracle is used to determine the Point-tuple$_i = (T_{i1}, \ldots, T_{it_i})$, with $T_{ik} = H_1(\mathrm{ID}_{i1}, \ldots, \mathrm{ID}_{ik})$ for $1 \le k \le t_i$, corresponding to ID-tuple$_i = (\mathrm{ID}_{i1}, \ldots, \mathrm{ID}_{it_i})$. In responding to these queries, algorithm $\mathcal{B}$ maintains a list H_1^{list} containing tuples of the form (ID-tuple$_i$, Point-tuple$_i$, Scalar-tuple$_i$, Secret-tuple$_i$). This list is initially empty. Algorithm $\mathcal{B}$ first adds ID-tuple$_0$ to H_1^{list} as follows:

For $1 \le k \le t_0$, algorithm $\mathcal{B}$:

1. picks a random $s_{0k} \in \mathbb{Z}/q\mathbb{Z}$;
2. picks a random $b_{0k} \in \mathbb{Z}/q\mathbb{Z}$;
3. sets $T_{01} = b_{01} P_1$ and $T_{0k} = b_{0k} P_0$ for $k > 1$.

Algorithm $\mathcal{B}$ then puts $((\mathrm{ID}_{01},\ldots,\mathrm{ID}_{0t_0}),(T_{01},\ldots,T_{0t_0}),(b_{01},\ldots,b_{0t_0}),s_{01},\ldots,s_{0t_0}))$ in H_1^{list}. When $\mathcal{A}$ queries H_1 about ID-tuple$_i$ =(ID$_{i1}$,…,ID$_{it_i}$), algorithm $\mathcal{B}$ responds as follows:

Let y be maximal such that $(\mathrm{ID}_{i1},\ldots,\mathrm{ID}_{iy}) = (\mathrm{ID}_{j1},\ldots,\mathrm{ID}_{jy})$ for some tuple (ID-tuple$_j$, Point-tuple$_j$, Scalar-tuple$_j$, Secret-tuple$_j$) already in H_1^{list}. Let $w \le y$ be maximal such that $(\mathrm{ID}_{i1},\ldots,\mathrm{ID}_{iw}) = (\mathrm{ID}_{01},\ldots,\mathrm{ID}_{0w})$. Then:

1. For $1 \le k \le y$, $\mathcal{B}$ sets $T_{ik} = T_{jk}$, $s_{ik} = s_{jk}$, and $b_{ik} = b_{jk}$. (Note: this is independent of j.)
2. If $0 < w = y < t_i$, then for $y < k \le t_i$, algorithm $\mathcal{B}$:
   (a) picks a random $s_{ik} \in \mathbb{Z}/q\mathbb{Z}$;
   (b) picks a random $b_{ik} \in \mathbb{Z}/q\mathbb{Z}$;
   (c) sets $T_{i(w+1)} = b_{i(w+1)}P_0 - s_{iw}^{-1}b_{i1}P_1$ and sets $T_{ik} = b_{ik}P_0$ if $y+1 < k \le t_i$.
3. If $w < y$ or $w = 0$, then for $y < k \le t_i$, algorithm $\mathcal{B}$:
   (a) picks a random $s_{ik} \in \mathbb{Z}/q\mathbb{Z}$;
   (b) picks a random $b_{ik} \in \mathbb{Z}/q\mathbb{Z}$;
   (c) sets $T_{ik} = b_{ik}P_0$.

Algorithm $\mathcal{B}$ then puts $((\mathrm{ID}_{i1},\ldots,\mathrm{ID}_{it_i}),(T_{i1},\ldots,T_{it_i}),(b_{i1},\ldots,b_{it_i}),(s_{i1},\ldots,s_{it_i}))$ in H_1^{list} and returns $(T_{i1},\ldots,T_{it_i})$ to $\mathcal{A}$. Note that T_{ik} is always chosen uniformly in $\mathbb{G}_1$ and is independent of $\mathcal{A}$'s view as required.

Extraction Queries: At any time, algorithm $\mathcal{A}$ may make a private key extraction query on any ID-tuple$_i$ other than ID-tuple$_0$ and its ancestors. Algorithm $\mathcal{B}$ responds to this query as follows:

1. Run the above algorithm for responding to H_1-queries to obtain the appropriate tuple (ID-tuple$_i$, Point-tuple$_i$, Scalar-tuple$_i$, Secret-tuple$_i$) in H_1^{list}.
2. Define
$$S_{it_i} = \sum_{k=1}^{t_i} s_{i(k-1)}s_0T_{ik}$$
   where $s_{i0} := 1$ for all i. This is the private point. Algorithm $\mathcal{B}$ also gives $\mathcal{A}$ the points $\{Q_{ik} = s_{ik}Q_0 : 1 \le k \le t_i - 1\}$. These are the Q-values corresponding to the private point.

We leave it to the reader to verify that this is a valid private key for ID-tuple$_i$ that is always computable by $\mathcal{B}$. Note that $\mathcal{B}$ does not know s_0 or s_0P_1. That S_{it_i} is computable by $\mathcal{B}$ follows from the definition of $T_{i(w+1)}$ above.

Challenge: At any time, algorithm $\mathcal{A}$ may request a challenge ciphertext from $\mathcal{B}$ on ID-tuple$_0$. Let $C = [U, V]$ be the challenge ciphertext given to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ sets the BasicHIDE ciphertext C' to be $[b_{01}^{-1}U, b_{01}^{-1}b_{02}U,\ldots,b_{01}^{-1}b_{0t_0}U, V]$ and gives C' as a challenge to Algorithm $\mathcal{A}$. Note that C' is a BasicHIDE encryption of M under ID-tuple$_0$ as required. To see this, first observe that a valid private key for ID-tuple$_0$ has the form $S' = s_0T_{01} + \sum_{k=2}^{t_0} s'_{k-1}T_{0k}$ — along with the additional information $\{s'_{k-1}P_0 : 2 \le k \le t_0\}$ — for some set $\{s'_{k-1} : 2 \le k \le t_0\}$. Second, observe that:

$$\frac{\hat{e}(b_{01}^{-1}U, S)}{\prod_{k=2}^{t_0}\hat{e}(b_{01}^{-1}b_{0k}U, s'_{k-1}P_0)} = \hat{e}(b_{01}^{-1}U, s_0T_{01}) = \hat{e}(U, s_0P_1)\ .$$

Since the values of the above pairings are the same, the correct decryption of C' is M. (Recall that $\hat{e}$ is symmetric.)

Guess: Eventually, algorithm $\mathcal{A}$ will produce a guess M'. Algorithm $\mathcal{B}$ outputs M' as its guess for the decryption of C.

Claim: Algorithm $\mathcal{A}$'s view is identical to its view in the real attack. Furthermore, $\Pr[M = M'] \geq \epsilon$. The probability is over the random bits used by $\mathcal{A}$, $\mathcal{B}$ and the challenger.
Proof of Claim: All responses to H_1-queries are as in the real attack since each response is uniformly and independently distributed in $\mathbb{G}_1$. All responses to private key extraction queries are valid. Finally, the challenge ciphertext C' given to $\mathcal{A}$ is the BasicHIDE encryption of the random plaintext M under the ID-tuple chosen by $\mathcal{A}$. Therefore, by the definition of algorithm $\mathcal{A}$, it will output $M' = M$ with probability at least ϵ.

Extensions of this Security Proof: We note that the above NHID-OWE security proof can be extended in two ways. Suppose we allow Algorithm $\mathcal{A}$ to choose a set $\mathcal{S}$ of potential target nodes, rather than restricting $\mathcal{A}$ to a single target node. For convenience, we stipulate that $\mathcal{S}$ must contain all ancestors of nodes in $\mathcal{S}$. Consider two cases: 1) Algorithm $\mathcal{A}$ may query private keys only for nodes outside of $\mathcal{S}$; 2) Algorithm $\mathcal{A}$ is permitted to query private keys for nodes in $\mathcal{S}$ (other the challenge node and its ancestors). In the former case, the reduction remains perfect: if $\mathcal{A}$ succeeds with probability ϵ, then so does $\mathcal{B}$. In the latter case, it is easy to prove that if $\mathcal{A}$ succeeds with probability ϵ, $\mathcal{B}$ succeeds with probability at least $\epsilon/|\mathcal{S}|$. Thus, if $|\mathcal{S}|$ is polynomial in the security parameter — certainly a reasonable assumption if the adversary wants to target an *existing* node that belongs to someone — then the reduction is polynomial.

A.4 HIDE: Adaptively Chosen Target

The proof of Lemma 2, which is used to prove Theorem 2, is modeled after the proof of Lemma 4.2 of [4]. The difference is that, rather than one coin flip, one must flip t coins, where t is the level of the identity that is ultimately targeted.

Lemma 2. *Suppose that $\mathcal{A}$ is an HID-OWE adversary that makes at most $q_E > 0$ private key extraction queries and has advantage ϵ_t of successfully targeting a BasicHIDE node in Level$_t$, and suppose that the hash function H_1 is a random oracle. Then there is an OWE adversary $\mathcal{B}$ that has advantage at least $\epsilon_t(t/e(q_E + t))^t$ against BasicPub and running time $O(time(\mathcal{A}))$.*

Proof: We show how to construct an OWE adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage $\epsilon_t(t/e(q_E+t))^t$ against BasicPub. The game between the challenger and the adversary $\mathcal{B}$ starts with the challenger first generating a random public key by running the key generation algorithm of BasicPub. The result is a public key $K_{pub} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, P_1, H_2)$ and a private key $S_1 = s_0 P_1$. The challenger then picks a random plaintext $M \in \mathcal{M}$ and encrypts M using the encryption algorithm of BasicPub. It gives K_{pub} and the resulting ciphertext $C = [U, V]$ to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ is supposed to output a guess for M. Algorithm $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:
Setup: Algorithm $\mathcal{B}$ gives algorithm $\mathcal{A}$ the BasicHIDE system parameters $params = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P_0, Q_0, H_1, H_2)$. Here, H_1 is a random oracle controlled by $\mathcal{B}$.
H_1-queries: At any time, algorithm $\mathcal{A}$ can query the random oracle H_1. Essentially, this oracle will be used to determine the Point-tuple$_i$ $= (T_{i1}, \ldots, T_{it_i})$, with $T_{ik} = H_1(\mathrm{ID}_{i1}, \ldots, \mathrm{ID}_{ik})$ for $1 \leq k \leq t_i$, corresponding to an ID-tuple$_i$ $= (\mathrm{ID}_{i1}, \ldots, \mathrm{ID}_{it_i})$. In responding to these queries, algorithm $\mathcal{B}$ maintains a list H_1^{list} containing tuples of the form (ID-tuple$_i$, Point-tuple$_i$, Scalar-tuple$_i$, Secret-tuple$_i$, Coin-tuple$_i$). This list is initially empty. When $\mathcal{A}$ queries H_1 about ID-tuple$_i$, algorithm $\mathcal{B}$ responds as follows:
Let y be maximal such that $(\mathrm{ID}_{i1}, \ldots, \mathrm{ID}_{iy}) = (\mathrm{ID}_{j1}, \ldots, \mathrm{ID}_{jy})$ for some tuple $((\mathrm{ID}_{j1}, \ldots, \mathrm{ID}_{jt_j}), (T_{j1}, \ldots, T_{jt_j}), (b_{j1}, \ldots, b_{jt_j}), (s_{j1}, \ldots, s_{jt_j}), (c_{j1}, \ldots, c_{jt_j}))$ already in H_1^{list}. Then:

1. For $1 \leq k \leq y$, $\mathcal{B}$ sets $T_{ik} = T_{jk}$, $s_{ik} = s_{jk}$, $b_{ik} = b_{jk}$, and $c_{ik} = c_{jk}$. (Note that this is independent of j.)
2. For $y < k \leq t_i$, algorithm $\mathcal{B}$:
   (a) picks a random $s_{ik} \in \mathbb{Z}/q\mathbb{Z}$;
   (b) picks a random $b_{ik} \in \mathbb{Z}/q\mathbb{Z}$;
   (c) sets $c_{i0} = 0$ and generates a random coin $c_{ik} \in \{0, 1\}$ so that $\Pr[c_{ik} = 0] = \delta$ for some δ that will be determined later;
   (d) sets $T_{ik} = b_{ik}P_0$ if $c_{ik} = c_{i(k-1)}$;
   (e) sets $T_{ik} = b_{ik}P_1$ if $c_{ik} = 1$ and $c_{i(k-1)} = 0$;
   (f) sets $T_{ik} = b_{ik}P_0 - s_{i(k-1)}^{-1} b_{if(k)}P_1$ if $c_{ik} = 0$ and $c_{i(k-1)} = 1$, where $f(k) < k$ is the largest subscript for which $c_{if(k)} = 1$ and $c_{i(f(k)-1)} = 0$, where $s_{i(k-1)}^{-1}$ is the inverse of $s_{i(k-1)}$ modulo q.

Algorithm $\mathcal{B}$ then puts $((\mathrm{ID}_{i1}, \ldots, \mathrm{ID}_{it_i}), (T_{i1}, \ldots, T_{it_i}), (b_{i1}, \ldots, b_{it_i}), (s_{i1}, \ldots, s_{it_i}), (c_{i1}, \ldots, c_{it_i}))$ in H_1^{list} and returns $(T_{i1}, \ldots, T_{it_i})$ to $\mathcal{A}$. Note that T_{ik} is always chosen uniformly in $\mathbb{G}_1$ and is independent of $\mathcal{A}$'s view, as required.

Phase 1 Extraction Queries: Let ID-tuple$_i$ be a private key extraction query issued by algorithm $\mathcal{A}$. Algorithm $\mathcal{B}$ responds to this query as follows:

1. Run the above algorithm for responding to H_1-queries to obtain the appropriate tuple (ID-tuple$_i$, Point-tuple$_i$, Scalar-tuple$_i$, Secret-tuple$_i$, Coin-tuple$_i$) in H_1^{list}.
2. If $c_{it_i} = 1$, then $\mathcal{B}$ reports failure and terminates. The attack on BasicPub failed.
3. Otherwise, define $p_{i(k-1)} = s_{i(k-1)} s_{i(f(k)-1)} s_{i(f(f(k))-1)} \cdots$, where $s_{i0} = s_0$, and $s_{i(f(j)-1)} = 1$ if $f(j)$ does not exist for the given j. Also, define
$$S_{it_i} = \sum_{k=1}^{t_i} p_{i(k-1)} T_{ik} \ .$$
   This is the private point. Algorithm $\mathcal{B}$ also gives $\mathcal{A}$ the points $\{Q_{i(k-1)} = p_{i(k-1)}P_0 : 2 \leq k \leq t_i\}$. These are the Q-values corresponding to the private point.

We leave it to the reader to verify that this is a valid private key for ID-tuple$_i$ that is always computable by $\mathcal{B}$ when $c_{it_i} = 0$. Note that $\mathcal{B}$ does not know s_0 or s_0P_1, but does know s_0P_0 ($= Q_0$). That S_{it_i} is computable by $\mathcal{B}$ can be shown inductively from the definition of T_{ik}. (The definition of T_{ik} when $c_{ik} = 0$ and $c_{i(k-1)} = 1$ is designed to cause certain cancellations to occur in the computation of S_{ik}, so that this point can be computed by $\mathcal{B}$ even though $S_{i(k-1)}$ cannot.)

Challenge: Once algorithm $\mathcal{A}$ decides that Phase 1 is over it outputs an ID-tuple ($\mathrm{ID}_1, \ldots, \mathrm{ID}_t$) on which it wishes to be challenged. Algorithm $\mathcal{B}$ responds as follows:

1. Run the above algorithm for responding to H_1-queries to obtain the appropriate tuple $((\mathrm{ID}_1, \ldots, \mathrm{ID}_t), (T_1, \ldots, T_t), (b_1, \ldots, b_t), (s_1, \ldots, s_t), (c_1, \ldots, c_t))$ in H_1^{list}. If $c_k = 0$ for some $1 \leq k \leq t$, then $\mathcal{B}$ reports failure and terminates. The attack on BasicPub failed.
2. Otherwise, we know $T_1 = b_1P_1$ and $T_k = b_kP_0$ for $2 \leq k \leq t$.
3. Let $C = [U, V]$ be the challenge ciphertext given to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ sets the BasicHIDE ciphertext C' to be $[b_1^{-1}U, b_1^{-1}b_2U, \ldots, b_1^{-1}b_tU, V]$. Algorithm $\mathcal{B}$ responds to $\mathcal{A}$ with the challenge C'.

Note that C' is a BasicHIDE encryption of M under the ID-tuple as required. To see this, first observe that a valid private key for ID-tuple$_i$ has the form $S'_t = s_0T_1 + \sum_{k=1}^{t-1} s'_k T_{k+1}$ (together with the additional information $\{s'_k P_0 : 1 \le k < t\})$ for some $(s'_1, \ldots, s'_{t-1}) \in (\mathbb{Z}/q\mathbb{Z})^{t-1}$. Second, observe that:

$$\frac{\hat{e}(b_1^{-1}U, S'_t)}{\prod_{k=2}^{t} \hat{e}(b_1^{-1} b_k U, s'_{k-1} P_0)} = \hat{e}(b_1^{-1}U, s_0 T_1) = \hat{e}(U, s_0 b_1^{-1} T_1) = \hat{e}(U, s_0 P_1) \ .$$

Since the values of the above pairings are the same, the correct decryption of C' is M. (Recall that $\hat{e}$ is symmetric.)

Phase 2: Algorithm $\mathcal{B}$ responds to private key extraction queries the way it did in Phase 1.

Guess: Eventually, algorithm $\mathcal{A}$ will produce a guess M'. Algorithm $\mathcal{B}$ outputs M' as its guess for the decryption of C.

Claim: If algorithm $\mathcal{B}$ does not abort during the simulation, then algorithm $\mathcal{A}$'s view is identical to its view in the real attack. Furthermore, if $\mathcal{B}$ does not abort then $\Pr[M = M'] \ge \epsilon_t$. The probability is over the random bits used by $\mathcal{A}$, $\mathcal{B}$, and the challenger.

Proof of Claim: All responses to H_1-queries are as in the real attack since each response is uniformly and independently distributed in $\mathbb{G}_1$. All responses to private key extraction queries are valid. Finally, the challenge ciphertext C' given to $\mathcal{A}$ is the BasicHIDE encryption of the random plaintext M under the ID-tuple chosen by $\mathcal{A}$. Therefore, by the definition of algorithm $\mathcal{A}$, it will output $M' = M$ with probability at least ϵ_t.

Probability: It remains to calculate the probability that $\mathcal{B}$ aborts during simulation. Suppose $\mathcal{A}$ makes a total of q_E private key extraction queries. The probability that $\mathcal{B}$ does not abort in phases 1 or 2 is δ^{q_E}, and the probability that it does not abort during the challenge step is $(1-\delta)^t$. Since these probabilities are independent, the overall probability that $\mathcal{B}$ does not abort is $\delta^{q_E}(1-\delta)^t$. The latter value is maximized at $\delta_{opt} = q_E/(q_E + t)$. Using δ_{opt}, the probability that $\mathcal{B}$ does not abort is at least $(t_i/e(q_E + t))^t$.

Remark 4. If $t = O(1)$ and q_E is polynomial in the security parameter, then $(t_i/e(q_E + t))^t$ is non-negligible in the security parameter, and we have a polynomial reduction from BasicPub to BasicHIDE (for adaptively-chosen-target adversaries).

A.5 Security Proofs for BasicHIDE and Dual-HIDE

The next result, which is Lemma 4.3 of [4], says that solving BDH reduces to breaking BasicPub.

Lemma 3. *Suppose that $\mathcal{A}$ is an OWE adversary with advantage ϵ against BasicPub that makes a total of q_{H_2} queries to the hash function H_2, and suppose that H_2 is a random oracle. Then there is an algorithm $\mathcal{B}$ that solves the BDH problem for $\mathcal{IG}$ with advantage at least $(\epsilon - \frac{1}{2^n})/q_{H_2}$ and running time $O(time(A))$.*

Theorem 1 for BasicHIDE follows by combining Lemmas 1 and 3. Theorem 2 for BasicHIDE follows by combining Lemmas 2 and 3. The proof of Theorem 1 for Dual-HIDE is the same as for BasicHIDE, except that algorithms $\mathcal{B}$ and $\mathcal{A}$ agree on both an ID-tuple that $\mathcal{A}$ will attack and a Sender-tuple that will send an encrypted message to that identity using Dual-HIDE. If the lowest-level common ancestor of the ID-tuple and the Sender-tuple is in Level$_m$, then $m-1$ is viewed as the root. Algorithm $\mathcal{B}$ sets $T_{0k} = b_{0k}P_1$ if $k = m$ and $T_{0k} = b_{0k}P_0$ otherwise, sets $T_{ik} = b_{ik}P_0$ if $w < m$ and $y < k \le t_i$, and sets $T_{i(w+1)} = b_{i(w+1)}P_0 - s_{iw}^{-1} b_{0m} P_1$ if $y = w < t_i$ and $w \ge m$. Algorithm $\mathcal{A}$ may make private key extraction queries on any ID-tuple, other than the one it is attacking and its ancestors in Level$_m$ or lower.

B Dual-HIDE with One Fewer Pairing Computation

As mentioned in Section 5.1, it is possible to decrease by one the number of values of the pairing that the encrypter must compute in Dual-HIDE. Here, each PKG E_t has a fixed random secret $s_t = \mathbb{Z}/q\mathbb{Z}$ that it uses, along with their identity points, to construct the private keys for its children, rather than generating a different random secret for each child. Encryption and decryption proceed as follows:

Encryption: To encrypt $M \in \mathcal{M}$, user y:

1. Computes $P_{zi} = H_1(\mathrm{ID}_{z1}, \ldots, \mathrm{ID}_{zi}) \in \mathbb{G}_1$ for $l + 1 \leq i \leq n$.
2. Chooses a random $r \in \mathbb{Z}/q\mathbb{Z}$.
3. Sets the ciphertext to be:

$$C = [rP_0, r(P_{y(l+1)} - P_{z(l+1)}), rP_{z(l+2)}, \ldots, rP_{zn}, M \oplus H_2(g^r_{y(l+1)})]$$
$$\text{where } g_{y(l+1)} = \frac{\hat{e}(P_0, S_y)}{\prod_{i=l+2}^{m} \hat{e}(Q_{y(i-1)}, P_{yi})} = \hat{e}(P_0, S_{y(l+1)}) \ ,$$

where S_y is y's secret point and $S_{y(l+1)}$ is the secret point of y's ancestor at level $l + 1$.

Decryption: Let $C = [U_0, U_{l+1}, \ldots, U_n, V]$ be the ciphertext. To decrypt C, user z computes:

$$V \oplus H_2(\frac{\hat{e}(U_0, S_z)\hat{e}(U_{l+1}, Q_{zl})}{\prod_{i=l+2}^{n} \hat{e}(Q_{z(i-1)}, U_i)}) = M.$$

C An alternative HIDS Scheme

Next we modify the HIDS scheme in Section 4 so that an entity's point-tuple $(P_1, \ldots, P_t)$ is computed as a function not only of its ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$, but also of the points $Q_i = s_iP_0$ for $1 \leq i \leq t$, where s_i is the secret of the entity's ancestor in Level_i. This commits the entity to its entire chain of secrets. However, this HIDS scheme is not very compatible with HIDE, because an encrypter would not know the entity's Q-values without an online lookup.

Root Setup: As in the HIDS scheme in Section 4.1.

Lower-level Setup: Entity $\mathrm{E}_t \in \mathrm{Level}_t$ picks a random $s_t \in \mathbb{Z}/q\mathbb{Z}$ and computes $Q_t = s_tP_0$.

Extraction: Let E_t be an entity in Level_t with ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$, where $(\mathrm{ID}_1, \ldots, \mathrm{ID}_i)$ for $1 \leq i \leq t$ is the ID-tuple of E_t's ancestor at Level_i. Set S_0 to be the identity element of $\mathbb{G}_1$. First, E_t sends Q_t (or alternatively, s_t) to its parent. Then E_t's parent:

1. computes $P_t = H_1(\mathrm{ID}_1, \ldots, \mathrm{ID}_t, Q_1, \ldots, Q_t) \in \mathbb{G}_1$;
2. sets E_t's secret point S_t to be $S_{t-1} + s_{t-1}P_t = \sum_{i=1}^{t} s_{i-1}P_i$;
3. also gives E_t the values $Q_i = s_iP_0$ for $1 \leq i \leq t - 1$.

Signing: To sign M with ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_t)$ (using the secret point S_t and the points $Q_i = s_iP_0$ for $1 \leq i \leq t$) do the following:

1. Compute $P_M = H_3(\mathrm{ID}_1, \ldots, \mathrm{ID}_t, M, Q_1, \ldots Q_t) \in \mathbb{G}_1$. (Again, H_3 need not be a totally different hash function.)
2. Compute $Sig(\text{ID-tuple}, M) = S_t + s_tP_M$, and send it along with $\{Q_i : 1 \leq i \leq t\}$.

Verification: Let $[Sig, Q_1, \ldots, Q_t] \in \mathcal{S}$ be the signature for $(\text{ID-tuple}, M)$. The verifier computes $P_i = H_1(\text{ID}_1, \ldots, \text{ID}_i, Q_1, \ldots, Q_i)$ for $1 \leq i \leq t$ and $P_M = H_1(\text{ID}_1, \ldots, \text{ID}_t, M, Q_1, \ldots Q_t)$ and confirms that:

$$\hat{e}(P_0, Sig) = \hat{e}(Q_0, P_1)\hat{e}(Q_t, P_M) \prod_{i=2}^{t} \hat{e}(Q_{i-1}, P_i).$$