A Forward-Secure Public-Key Encryption Scheme*

Ran Canetti† Shai Halevi† Jonathan Katz‡

Abstract

Cryptographic computations are often carried out on insecure devices for which the threat of key exposure represents a serious concern. *Forward security* allows one to mitigate the damage caused by exposure of secret keys. In a forward-secure scheme, secret keys are updated at regular periods of time; exposure of the secret key corresponding to a given time period does not enable an adversary to "break" the scheme (in the appropriate sense) for any *prior* time period.

We present the first constructions of (non-interactive) forward-secure public-key encryption schemes. Our main construction achieves security against chosen-plaintext attacks in the standard model, and all parameters of the scheme are poly-logarithmic in the total number of time periods. Some variants and extensions of this scheme are also given.

We also introduce the notion of *binary tree encryption* and construct a binary tree encryption scheme in the standard model. Our construction implies the first (hierarchical) identity-based encryption scheme in the standard model. (The notion of security we achieve, however, is slightly weaker than that achieved by some previous constructions in the random oracle model.)

1 Introduction

Exposure of secret keys can be a devastating attack on a cryptosystem since such an attack typically implies that all security guarantees are lost. Indeed, standard notions of security offer no protection whatsoever once the secret key of the system has been compromised. With the threat of key exposure becoming more acute as cryptographic computations are performed more frequently on poorly-protected devices (smart-cards, mobile phones, etc.), new techniques are needed to deal with this concern.

A variety of methods, including secret sharing [38], threshold cryptography [16], and proactive cryptography [34], have been introduced in an attempt to deal with this threat. One promising approach — which we focus on here — is to construct *forward-secure* cryptosystems. This notion was first proposed in the context of key-exchange protocols by Günther [25] and Diffie, et al. [17]: a forward-secure key-exchange protocol guarantees that exposure of long-term secret information does not compromise the security of previously-generated session keys. A forward-secure key-exchange protocol naturally gives rise to a forward-secure *interactive* encryption scheme in which the sender and receiver interact to generate a shared key which is erased immediately after being used to encrypt a single message.

Subsequently, Anderson [3] suggested forward security for the more challenging *non-interactive* setting: here, the lifetime of the system is divided into N intervals (or time periods) labeled $0, \ldots, N-1$, and the secret key "evolves" with time. Namely, at the beginning of time period i any

*A preliminary version of this work appeared in [12].

†IBM T.J. Watson Research Center, NY, USA. {canetti,shaih}@watson.ibm.com.

‡Dept. of Computer Science, University of Maryland. Portions of this work were done while at Columbia University.
Work supported in part by NSF Trusted Computing Grant #0310751. jkatz@cs.umd.edu.

party who stores the secret key applies some function to the "previous" key SK_{i-1} to derive the "current" key SK_i; key SK_{i-1} is then *erased* and SK_i is used for all secret cryptographic operations during period i. If we are in a public-key setting, the public key remains *fixed* throughout the lifetime of the system; this is crucial for making the scheme viable. Forward security means that exposure of the secret key SK_i (for any time period i) does not compromise the security of the system — in some appropriate sense — for *all* time periods *prior* to i. (Note that since SK_i is the only secret existing at period i, it is impossible to ensure security for period i or any subsequent time period in this model.) Specializing for the case of encryption, which is the focus of this work, forward security guarantees that even if an adversary learns SK_i (for some i), messages encrypted during all time periods *prior* to i remain secret. The notion of forward security was first formalized by Bellare and Miner [5] in the context of signature schemes; a formal definition for the case of public-key encryption is introduced here and given in Section 4.

A number of constructions of forward-secure signature/identification schemes are known [5, 30, 1, 27, 31, 29], and forward security in the symmetric-key setting has also been studied [6]. The existence of non-trivial, forward-secure public-key encryption (PKE) schemes, however, has been open since the question was first posed by Anderson [3]. Forward-secure PKE has a number of obvious applications, as it can be used to protect (to the extent possible) the secrecy of communications for devices operating in insecure environments where key exposure is an immediate concern. Of course, it is appropriate in "standard" environments as well: if used to send encrypted e-mail, for example, then the compromise of a user's secret key on a particular day does not leak any information about e-mails sent to that user at any time in the past. (Note, however, that if the user wants to retain the ability to decrypt past e-mails then he will have to store the "master" secret key SK_0 on some secure device.) Finally, forward-secure PKE forms an integral building block in recent constructions of *adaptively-secure* encryption schemes [14].

1.1 Our Contributions

In this work we construct the first (non-interactive) forward-secure public-key encryption schemes. Toward this goal, we introduce the notion of *binary tree encryption* and show a construction of the latter as well. Interestingly, this yields the first construction of a hierarchical identity-based encryption scheme that does not rely on the random oracle model. (The notion of security we achieve, however, is somewhat weaker than that achieved in prior work.) We explain these contributions in more detail now.

Forward-secure encryption. We formally define a notion of security for forward-secure public-key encryption and give efficient constructions of schemes satisfying this notion. Our main scheme achieves semantic security (i.e., security against chosen-plaintext attacks) in the standard model based on the decisional version of the bilinear Diffie-Hellman (BDH) assumption [28, 9]. All salient parameters of this scheme are poly-logarithmic in N, the total number of time periods.

We also present a variant of this scheme with better complexity: in particular, the public-key size and the key-generation/key-update times are independent of N. Here, semantic security is proven in the random oracle model[1] under the *computational* BDH assumption. The parameters of our schemes are summarized in Table 1. Both schemes are roughly as efficient as $\log_2 N$ invocations of the Boneh-Franklin identity-based encryption scheme [9] and are therefore practical for reasonable values of N.

[1] A proof in the random oracle model does not guarantee the security of a protocol once the random oracle is instantiated with an efficiently-computable "cryptographic hash function" [11]. Nevertheless, a proof in the random oracle model can be regarded as heuristic evidence that a construction is secure.

	Standard model	Random oracle model
Key generation time	$\mathcal{O}(\log N)$	$\mathcal{O}(1)$
Encryption/decryption time	$\mathcal{O}(\log N \cdot (\log \log N)^2)$	$\mathcal{O}(\log N)$
Key update time	$\mathcal{O}(\log N)$	$\mathcal{O}(1)$
Ciphertext length	$\mathcal{O}(\log N)$	$\mathcal{O}(\log N)$
Public key size	$\mathcal{O}(\log N)$	$\mathcal{O}(1)$
Secret key size	$\mathcal{O}(\log N)$	$\mathcal{O}(\log N)$

Table 1: Efficiency of our forward-secure encryption schemes as a function of the total number of time periods N.

At a high level, our constructions share similarities with previous tree-based, forward-secure signature schemes (e.g., those of [5, 1, 31]). Here, however, we associate time periods with *all* the nodes of the tree (in a pre-order traversal) instead of associating time periods with the leaves only; this improves the efficiency of our key-generation and key-update algorithms. This tree traversal technique can also be used to improve the efficiency of key generation and the (worst-case) efficiency of key updates in the tree-based signature schemes mentioned above, from $\mathcal{O}(\log N)$ to $\mathcal{O}(1)$.

We consider also a number of extensions of our schemes. Using the techniques of Malkin, et al. [31], our schemes can be adapted to support an unbounded number of time periods; in other words, the number of time periods N need not be known at the time the public key is generated and published. This has the added advantage that the efficiency depends only on the number of time periods elapsed thus far. We also sketch two ways to modify our schemes to achieve security against adaptive chosen-ciphertext attacks [35, 4]. In the random oracle model, we use (an appropriate modification of) the Fujisaki-Okamoto transformation [20]. In the standard model, we note that the techniques of Sahai [37] using simulation-sound NIZK proofs (and based on earlier work of Naor and Yung [33]) extend to our setting; interestingly, we show also that NIZK proofs for all of $\mathcal{NP}$ may be constructed based on the computational BDH assumption (so that we do not require the additional assumption of trapdoor permutations). This approach serves as a proof of feasibility only, as it results in a very inefficient scheme. Subsequent to our work, more efficient methods for achieving chosen-ciphertext security in our setting were shown [13, 10].

Binary-tree encryption and (hierarchical) identity-based encryption. Our constructions are based on the hierarchical identity-based encryption (HIBE) scheme of Gentry and Silverberg [21] which, in turn, is based on the identity-based encryption (IBE) scheme of Boneh and Franklin [9]. As a first step toward our constructions, we define a relaxed variant of HIBE which we call *binary tree encryption* (BTE). We then show how to modify the Gentry-Silverberg construction to yield a BTE scheme which can be proven secure *in the standard model* for trees of *polynomial depth*. (In contrast, the main construction of Gentry and Silverberg is proven secure in the random oracle model, and only for trees of constant depth.) Finally, we construct a forward-secure encryption scheme from any BTE scheme. Our construction of a forward-secure encryption scheme can be slightly optimized when given a HIBE scheme (rather than a BTE scheme) as a primitive; as an example, a more efficient forward-secure encryption scheme can be constructed using a recent HIBE scheme of Boneh, et al. [8].

The BTE primitive is interesting in its own right. We show in Section 5 how a full-blown IBE/HIBE scheme (albeit satisfying a slightly weaker notion of security than that considered by Boneh-Franklin and Gentry-Silverberg) may be based on any BTE scheme. Combined with our construction of a BTE scheme, this yields the first (hierarchical) identity-based encryption scheme

with a proof of security in the standard model.

1.2 Organization

In Section 2 we define the computational and decisional versions of the BDH assumption, and also review the notion of t-wise independent function families as needed in this work. In Section 3 we define binary tree encryption and provide a construction of a BTE scheme which is provably secure under the decisional BDH assumption in the standard model. In that section we also show a more efficient construction based on the computational BDH assumption in the random oracle model and discuss some extensions of our schemes.

We formally define forward security for public-key encryption in Section 4, and show there how a forward-secure PKE scheme can be constructed from any BTE scheme. Combining our results, we obtain a forward-secure PKE scheme with the parameters advertised in Table 1. In Section 5 we define a (slightly) relaxed notion of security for hierarchical identity-based encryption, and show how an HIBE scheme satisfying this notion can be constructed from any BTE scheme. Combining this with our results from Section 3 yields an HIBE scheme secure in the standard model.

2 Preliminaries

We let PPT stand for "probabilistic polynomial time." If A is a probabilistic algorithm taking inputs $x_1, \ldots, x_n$, then by $y = A(x_1, \ldots, x_n; \omega)$ we mean that y is assigned the (deterministic) output of A when run on the stated inputs with random coins ω. By $y \leftarrow A(x_1, \ldots, x_n)$ we mean that random coins ω are chosen uniformly at random, and y is assigned the value $A(x_1, \ldots, x_n; \omega)$.

Let ε denote the empty string, having length 0. We let $\{0,1\}^\ell$ denote the set of strings of length ℓ, and define $\{0,1\}^{<\ell} \stackrel{\text{def}}{=} \bigcup_{0 \le i < \ell} \{0,1\}^i$ and $\{0,1\}^{\le \ell} \stackrel{\text{def}}{=} \bigcup_{0 \le i \le \ell} \{0,1\}^i$. We stress in particular that the latter both contain the empty string.

2.1 The Bilinear Diffie-Hellman Assumption

The security of our BTE schemes are based on the difficulty of the bilinear Diffie-Hellman (BDH) problem as formalized by Boneh and Franklin [9] (see also [28]). We review the relevant definitions of the computational and decisional versions of this assumption as they appear in [9, 21]. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order q, where $\mathbb{G}_1$ is represented additively and $\mathbb{G}_2$ is represented multiplicatively. We assume a non-constant map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ for which the following hold:

1. The map $\hat{e}$ is *bilinear*: for all $P_0, P_1 \in \mathbb{G}_1$ and all $\alpha, \beta \in \mathbb{Z}_q$ we have $\hat{e}(\alpha P_0, \beta P_1) = \hat{e}(P_0, P_1)^{\alpha\beta}$.
2. There is an efficient algorithm to compute $\hat{e}(P_0, P_1)$ for any $P_0, P_1 \in \mathbb{G}_1$.

A *BDH parameter generator* $\mathcal{IG}$ is a randomized, polynomial-time algorithm that takes as input a security parameter 1^k and outputs the description of two groups $\mathbb{G}_1, \mathbb{G}_2$ and a map $\hat{e}$ satisfying the above conditions (we assume q, the group order, is implicit in $\mathbb{G}_1, \mathbb{G}_2$). We define the *computational BDH problem with respect to* $\mathcal{IG}$ as the following: given $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ output by $\mathcal{IG}$ along with random $P, \alpha P, \beta P, \gamma P \in \mathbb{G}_1$, compute $\hat{e}(P,P)^{\alpha\beta\gamma}$. We say that $\mathcal{IG}$ *satisfies the computational BDH assumption* if the following probability is negligible (in k) for all PPT algorithms A:

$$\Pr\left[\begin{array}{c}(\mathbb{G}_1, \mathbb{G}_2, \hat{e}) \leftarrow \mathcal{IG}(1^k); P \leftarrow \mathbb{G}_1; \alpha, \beta, \gamma \leftarrow \mathbb{Z}_q : \\ A(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, \alpha P, \beta P, \gamma P) = \hat{e}(P,P)^{\alpha\beta\gamma}\end{array}\right].$$

The *decisional BDH problem* is to distinguish between tuples of the form $(P, \alpha P, \beta P, \gamma P, \hat{e}(P,P)^{\alpha\beta\gamma})$ and $(P, \alpha P, \beta P, \gamma P, \hat{e}(P,P)^{\mu})$ for random $P, \alpha, \beta, \gamma, \mu$. (Note that if P is a generator of $\mathbb{G}_1$ — which is the case with all but negligible probability — then $\hat{e}(P,P)$ is a generator of $\mathbb{G}_2$ and so $\hat{e}(P,P)^{\mu}$ is simply a random element of $\mathbb{G}_2$.) Formally, we say $\mathcal{IG}$ *satisfies the decisional BDH assumption* if the following probability is negligible (in k) for all PPT algorithms A:

$$\left| \Pr\left[\begin{array}{l} (\mathbb{G}_1, \mathbb{G}_2, \hat{e}) \leftarrow \mathcal{IG}(1^k); P \leftarrow \mathbb{G}_1; \alpha, \beta, \gamma \leftarrow \mathbb{Z}_q : \\ A(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, \alpha P, \beta P, \gamma P, \hat{e}(P,P)^{\alpha\beta\gamma}) = 1 \end{array} \right] \right.$$
$$\left. - \Pr\left[\begin{array}{l} (\mathbb{G}_1, \mathbb{G}_2, \hat{e}) \leftarrow \mathcal{IG}(1^k); P \leftarrow \mathbb{G}_1; \alpha, \beta, \gamma, \mu \leftarrow \mathbb{Z}_q : \\ A(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, \alpha P, \beta P, \gamma P, \hat{e}(P,P)^{\mu}) = 1 \end{array} \right] \right| .$$

BDH parameter generators believed to satisfy the above assumptions can be constructed from modified Weil or Tate pairings associated with elliptic curves or Abelian varieties. As our results do not depend on any specific instantiation, we refer the interested reader to [9] for details.

2.2 t-Wise Independent Function Families

We briefly review the notion of t-wise independent function families (specialized for our purposes) and describe the construction we will use. Let $\mathcal{H}$ be a family of functions with domain $\mathbb{Z}_q$ and range $\mathbb{G}_1$ (where these are as in the previous section; in particular, q is prime). We say $\mathcal{H}$ is *t-wise independent* if for all distinct $x_1, \ldots, x_t \in \mathbb{Z}_q$ and all $y_1, \ldots, y_t \in \mathbb{G}_1$ we have

$$\Pr_{H \leftarrow \mathcal{H}}[H(x_1) = y_1 \wedge \cdots \wedge H(x_t) = y_t] = \left(\frac{1}{|\mathbb{G}_1|}\right)^t .$$

In other words, informally speaking, any t distinct elements in $\mathbb{Z}_q$ are mapped uniformly and independently to $\mathbb{G}_1$ by a randomly-selected hash function from $\mathcal{H}$. Abusing terminology somewhat, we will refer to a given function $H \in \mathcal{H}$ as a *t-wise independent function*. An additional property we will require of $\mathcal{H}$ is that given any distinct $x_1, \ldots, x_j \in \mathbb{Z}_q$ and any $y_1, \ldots, y_j \in \mathbb{G}_1$ with $j \leq t$, it is possible to efficiently sample a uniform element from the set

$$\{H \in \mathcal{H} : H(x_1) = y_1 \wedge \cdots \wedge H(x_j) = y_j\} .$$

We will use the following construction: let $\mathcal{H} = \{H_{h_0,\ldots,h_t}\}_{h_0,\ldots,h_t \in \mathbb{G}_1}$, where $H_{h_0,\ldots,h_t}(x) \stackrel{\text{def}}{=} h_0 + x \cdot h_1 + \cdots + x^t \cdot h_t$. We first claim that $\mathcal{H}$ is $(t+1)$-wise independent. To see this, let $g \in \mathbb{G}_1$ be a generator of $\mathbb{G}_1$ and let $\hat{H}_{h_0,\ldots,h_t}$ denote the polynomial (over the field $\mathbb{Z}_q$)

$$\hat{H}_{h_0,\ldots,h_t}(x) = \log_g h_0 + x \cdot \log_g h_1 + \cdots + x^t \cdot \log_g h_t ,$$

where $\log_g h$ is the unique element $\lambda \in \mathbb{Z}_q$ such that $\lambda \cdot g = h$ (recall that $|\mathbb{G}_1| = q$). Now, for any distinct $x_1, \ldots, x_{t+1} \in \mathbb{Z}_q$ and $y_1, \ldots, y_{t+1} \in \mathbb{G}_1$, we have

$$H_{h_0,\ldots,h_t}(x_i) = y_i \text{ for all } i \quad \text{iff} \quad \hat{H}_{h_0,\ldots,h_t}(x_i) = \log_g y_i \text{ for all } i.$$

It is well known that there is a *unique* polynomial $\hat{H}^*(x) = \hat{h}_0^* + x \cdot \hat{h}_1^* + \cdots + x^t \cdot \hat{h}_t^*$ of degree at most t such that $\hat{H}^*(x_i) = \log_g y_i$ for all i. So

$$\begin{aligned} \Pr_{H \leftarrow \mathcal{H}}[\forall i : H(x_i) = y_i] &= \Pr_{h_0,\ldots,h_t \leftarrow \mathbb{G}_1}[\forall i : H_{h_0,\ldots,h_t}(x_i) = y_i] \\ &= \Pr_{h_0,\ldots,h_t \leftarrow \mathbb{G}_1}[\hat{H}_{h_0,\ldots,h_t} = \hat{H}^*] \end{aligned}$$

$$= \Pr_{h_0,\ldots,h_t \leftarrow \mathbb{G}_1}[\forall i : \log_g h_i = \hat{h}_i^*]$$
$$= \left(\frac{1}{q}\right)^{t+1} = \left(\frac{1}{|\mathbb{G}_1|}\right)^{t+1} ,$$

as required.

As for our second requirement, given distinct $x_1, \ldots, x_j \in \mathbb{Z}_q$ and arbitrary $y_1, \ldots, y_j \in \mathbb{G}_1$ with $j \leq t+1$, we may sample uniformly from the set $\{H \in \mathcal{H} : H(x_i) = y_i \text{ for } 1 \leq i \leq j\}$ as follows. Choose arbitrary $x_{j+1}, \ldots, x_{t+1} \in \mathbb{Z}_q$ so that the $\{x_i\}_{i=1}^{t+1}$ are distinct. Then choose $y_{j+1}, \ldots, y_{t+1} \in \mathbb{G}_1$ uniformly at random. We now find the unique values $h_0, \ldots, h_t$ such that $H_{h_0,\ldots,h_t}(x_i) = y_i$ for all i. These values must satisfy the following system of equations:

$$\underbrace{\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^t \\ 1 & x_2 & x_2^2 & \cdots & x_2^t \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{t+1} & x_{t+1}^2 & \cdots & x_{t+1}^t \end{pmatrix}}_{X} \cdot \begin{pmatrix} h_0 \\ h_1 \\ \vdots \\ h_t \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{t+1} \end{pmatrix} .$$

Since the $\{x_i\}$ are distinct, the Vandermonde matrix X is invertible. We may thus compute

$$\begin{pmatrix} h_0 \\ h_1 \\ \vdots \\ h_t \end{pmatrix} = X^{-1} \cdot \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{t+1} \end{pmatrix} ,$$

as desired. Note in particular that we do not need to compute any discrete logarithms in $\mathbb{G}_1$ (which we do not know how to do efficiently when $\mathbb{G}_1$ is generated as in the previous section).

3 Binary Tree Encryption

This section defines the notion of binary tree encryption (BTE) and presents a BTE scheme based on the bilinear Diffie-Hellman assumption. As discussed in the introduction, BTE is a relaxation of hierarchical identity-based encryption (HIBE) [26, 21]. As in HIBE, in BTE we have a "master" public key PK associated with a tree; each node in this tree has a corresponding secret key. To encrypt a message "targeted" for some node, one uses both PK and the name of the target node; the resulting ciphertext can then be decrypted using the secret key of the target node. Moreover, as in HIBE the secret key of any node can be used to derive the secret keys for the children of that node. The only difference between HIBE and BTE is that in the latter we insist on a *binary* tree, where the children of a node w are labeled $w0$ and $w1$. (In an HIBE scheme the tree can have arbitrary degree, and a child of node v is labeled $v.s$ for an arbitrary string s.) A formal definition follows:

Definition 1 A (public-key) *binary tree encryption (BTE) scheme* is a tuple of PPT algorithms $(\mathsf{Gen}, \mathsf{Der}, \mathsf{Enc}, \mathsf{Dec})$ such that:

- The *key-generation algorithm* Gen takes as input a security parameter 1^k and a value 1^ℓ representing the depth of the tree. It returns a master public key PK and a root secret key SK_ε. (We assume that 1^k and 1^ℓ are implicit in PK.)

- The *key-derivation algorithm* Der takes as input PK, the name of a node $w \in \{0,1\}^{<\ell}$, and its secret key SK_w. It returns secret keys SK_{w0}, SK_{w1} for the two children of w.
- The *encryption algorithm* Enc takes as input PK, the name of a node $w \in \{0,1\}^{\leq \ell}$, and a message M. It returns a ciphertext C.
- The *decryption algorithm* Dec takes as input PK, the name of a node $w \in \{0,1\}^{\leq \ell}$, its secret key SK_w, and a ciphertext C. It returns a message M or a distinguished symbol $\perp$.

We make the natural correctness requirement: namely, for any (PK, SK_ε) output by $\mathsf{Gen}(1^k, 1^\ell)$, any node $w \in \{0,1\}^{\leq \ell}$ and secret key SK_w correctly generated for this node, and any message M, we have $M = \mathsf{Dec}(PK, w, SK_w, \mathsf{Enc}(PK, w, M))$. $\diamondsuit$

Roughly speaking, a secure BTE scheme should ensure the secrecy of ciphertexts targeted for a node w even if the secret keys of other nodes (as long as they are *not* ancestors of w) are exposed. In [21] (in the context of HIBE), the adversary is allowed to choose the target node w adaptively. We define a relaxed notion of security whereby the adversary is required to commit to the target node *in advance* (i.e., before seeing the public key); we call this attack scenario a *selective-node (SN) attack* (by analogy with "selective forgery" of signatures [24]). While this definition is a weaker one, it suffices for our applications. Furthermore, by a standard hybrid argument the definitions can be shown to be equivalent when the number of nodes in the tree is polynomial in the security parameter.

Definition 2 A BTE scheme is *secure against selective-node, chosen-plaintext attacks* (secure in the sense of $\mathsf{SN\text{-}CPA}$) if for all polynomials $\ell(\cdot)$ and all PPT adversaries A, the advantage of A in the following game is negligible in the security parameter k (in the following, let $\ell = \ell(k)$):

1. $A(1^k, 1^\ell)$ outputs a name $w^* \in \{0,1\}^{\leq \ell}$ of a node.
2. Algorithm $\mathsf{Gen}(1^k, 1^\ell)$ outputs (PK, SK_ε). In addition, algorithm $\mathsf{Der}(\cdots)$ is run to generate the secret keys of all the nodes on the path from the root to w^* (we denote this path by P). The adversary is given PK and the secret keys $\{SK_w\}$ for all nodes w of the following form:
   – $w = w'\overline{b}$, where $w'b$ is a prefix of w^* and $b \in \{0,1\}$ (i.e., w is a sibling of some node in P);
   – $w = w^*0$ and $w = w^*1$, if $|w^*| < \ell$ (i.e., w is a child of w^*).
   Note that this information allows the adversary to compute $SK_{w'}$ for any node $w' \in \{0,1\}^{\leq \ell}$ that is not a prefix of w^*.
3. The adversary generates a request $\mathsf{challenge}(M_0, M_1)$. A random bit b is selected and the adversary is given $C^* = \mathsf{Enc}(PK, w^*, M_b)$.

At the end of the game the adversary outputs $b' \in \{0,1\}$; it succeeds if $b' = b$. The adversary's *advantage* is the absolute value of the difference between its success probability and $1/2$. $\diamondsuit$

In the above definition (as well as all the definitions of security in this paper) the adversary is assumed to maintain state throughout its execution.

Security against chosen-ciphertext attacks is defined as the obvious extension of the above:

Definition 3 A BTE scheme is *secure against selective-node, chosen-ciphertext attacks* (secure in the sense of $\mathsf{SN\text{-}CCA}$) if for all polynomials $\ell(\cdot)$ and all PPT adversaries A, the advantage of A in the following game is negligible in the security parameter k (again, set $\ell = \ell(k)$):

1. $A(1^k, 1^\ell)$ outputs a name $w^* \in \{0,1\}^{\leq \ell}$ of a node.

2. Algorithm $\mathsf{Gen}(1^k, 1^\ell)$ outputs (PK, SK_ε). The adversary is given PK and node secret keys as in Definition 2.

3. The adversary may query a decryption oracle denoted by $\mathsf{Dec}^*(\cdot,\cdot)$. On query $\mathsf{Dec}^*(w, C)$ with $w \in \{0,1\}^{\leq \ell(k)}$, the key SK_w is derived from SK_ε and the adversary is given $M = \mathsf{Dec}(PK, w, SK_w, C)$.

4. The adversary generates a request $\mathsf{challenge}(M_0, M_1)$. A random bit b is selected and the adversary is given $C^* = \mathsf{Enc}(PK, w^*, M_b)$.

5. The adversary may continue to query $\mathsf{Dec}^*(\cdot,\cdot)$, except that it may not query $\mathsf{Dec}^*(w^*, C^*)$ (but it may query $\mathsf{Dec}^*(w, C^*)$ with $w \neq w^*$ or $\mathsf{Dec}^*(w^*, C)$ with $C \neq C^*$).

At the end of the game the adversary outputs $b' \in \{0,1\}$; it succeeds if $b' = b$. The adversary's *advantage* is the absolute value of the difference between its success probability and $1/2$. $\diamondsuit$

Remark 1 (Randomized key-derivation algorithms). There is a slight technicality with regard to the above definition in case the key-derivation algorithm Der is randomized (and so there might be multiple "valid" keys SK_w that can be derived from SK_ε). Specifically, there are two natural ways the decryption queries of A can be answered:

First approach: When A queries $\mathsf{Dec}^*(w, C)$, key SK_w is derived "from scratch" starting from SK_ε using repeated calls to Der.

Second approach: At the end of step 2, define a set Keys containing SK_ε, all secret keys on the path from the root to w^*, and all secret keys given to A. When A queries $\mathsf{Dec}^*(w, C)$, do:

- If $SK_w \in \mathsf{Keys}$, decrypt C using SK_w.
- Otherwise, let w' be the longest prefix of w such that $SK_{w'} \in \mathsf{Keys}$. Derive SK_w using $SK_{w'}$ and repeated calls to Der. Decrypt C using SK_w, and add all secret keys generated during this step to Keys.

Note that, under the first approach, the same decryption query of A might be answered differently depending on the secret key used for decryption each time the query is asked. For the schemes presented here, the choice of which approach to use is immaterial even though key derivation is randomized. For concreteness, however, we will implicitly assume the second approach.

3.1 BTE Schemes Based on the BDH Assumption

Our main result of this section is the following:

Theorem 1 *Under the decisional BDH assumption, there exists a BTE scheme that is secure in the sense of SN-CPA.*

The starting point for our construction is the HIBE scheme of Gentry and Silverberg [21]. Unlike their scheme, our scheme will be proven secure in the standard model and for trees of polynomial depth. (It is immediate that the scheme of [21] may be used to implement a secure BTE in the random oracle model for trees of constant depth.) The HIBE scheme of Gentry and Silverberg (as well as the IBE scheme of Boneh and Franklin [9]) uses random oracles in three ways: to map

identities to group elements, to efficiently achieve semantic security based on the computational BDH assumption, and to obtain chosen-ciphertext security. The latter two uses of the random oracle can be avoided if one is willing to rely on the *decisional* BDH assumption (to efficiently achieve semantic security) and generic non-interactive zero-knowledge (to achieve chosen-ciphertext security, as discussed further below). More interestingly, we show that the random oracle used to map identities to group elements can be replaced by an $\mathcal{O}(\ell)$-wise independent function (see Section 2.2), where ℓ is the depth of the tree. Moreover, a proof of security may be obtained even for trees of polynomial depth. We now provide the details.

Notation and conventions. Recall that ℓ denotes the depth of the tree. The i-bit prefix of a string $w_1 w_2 \cdots w_t$ is denoted by $w|_i$. Namely, $w|_i = w_1 \cdots w_i$. By convention, we set $w|_0 = \varepsilon$ (i.e., the empty string) for any string w. Let $\mathcal{H}_\ell$ denote a $(2\ell+1)$-wise independent family of functions with domain $\{0,1\}^{\leq \ell}$ and range $\mathbb{G}_1$; we may take essentially the construction described in Section 2.2 except that we first apply a one-to-one encoding of elements in $\{0,1\}^{\leq \ell}$ as elements in $\mathbb{Z}_q$ (alternately, we can include in the public key a universal one-way hash function [32] mapping $\{0,1\}^{\leq \ell}$ to $\mathbb{Z}_q$). Finally, $\mathcal{IG}$ is a BDH parameter generator.

$\mathsf{Gen}(1^k, 1^\ell)$ does the following:

1. Run $\mathcal{IG}(1^k)$ to generate groups $\mathbb{G}_1, \mathbb{G}_2$ (of prime order q) and bilinear map $\hat{e}$.
2. Select a random generator $P \in \mathbb{G}_1$ and a random $\alpha \in \mathbb{Z}_q$. Set $Q = \alpha P$.
3. Choose a random function $H \in \mathcal{H}_\ell$.
4. The public key is $PK = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, H)$. The root secret key is $SK_\varepsilon = \alpha H(\varepsilon)$.

In general, the secret key of node $w = w_1 \cdots w_t$ consists of $t+1$ group elements and is denoted by $SK_w = (R_{w|_0}, R_{w|_1}, \ldots, R_{w|_{t-1}}, S_w)$ (for the special case of $w = \varepsilon$ we simply have $SK_\varepsilon = S_\varepsilon = \alpha H(\varepsilon)$ and the other values are not present). With this in mind, we now describe the key derivation algorithm.

$\mathsf{Der}(PK, w, SK_w)$ does the following:

1. Let $w = w_1 \cdots w_t$. Parse SK_w as $(R_{w|_0}, R_{w|_1}, \ldots, R_{w|_{t-1}}, S_w)$.
2. Choose random $\rho_w \in \mathbb{Z}_q$. Set $R_w = \rho_w P$, $S_{w0} = S_w + \rho_w H(w0)$, and $S_{w1} = S_w + \rho_w H(w1)$.
3. Output $SK_{w0} = (R_{w|_0}, \ldots, R_w, S_{w0})$ and $SK_{w1} = (R_{w|_0}, \ldots, R_w, S_{w1})$.

$\mathsf{Enc}(PK, w, M)$ (where $M \in \mathbb{G}_2$) does the following:

1. Let $w = w_1 \cdots w_t$. Select random $\gamma \in \mathbb{Z}_q$.
2. Output $C = (\gamma P,\ \gamma H(w|_1),\ \gamma H(w|_2), \ldots, \gamma H(w),\ M \cdot d)$, where $d = \hat{e}(Q, H(\varepsilon))^\gamma$.

$\mathsf{Dec}(PK, w, SK_w, C)$ does the following:

1. Let $w = w_1 \cdots w_t$. Parse SK_w as $(R_{w|_0}, \ldots, R_{w|_{t-1}}, S_w)$ and parse C as $(U_0, U_1, \ldots, U_t, V)$.
2. Output $M = V/d$, where
$$d = \frac{\hat{e}(U_0, S_w)}{\prod_{i=1}^{t} \hat{e}(R_{w|_{i-1}}, U_i)}.$$

We verify that decryption succeeds. When encrypting, we have $d = \hat{e}(Q, H(\varepsilon))^{\gamma} = \hat{e}(P, H(\varepsilon))^{\alpha\gamma}$. When decrypting, we have $U_0 = \gamma P$, and $U_i = \gamma H(w|_i)$ for $1 \le i \le t$ (where $t = |w|$). Hence,

$$\begin{aligned} d &= \frac{\hat{e}(U_0, S_w)}{\prod_{i=1}^{t} \hat{e}(R_{w|_{i-1}}, U_i)} = \frac{\hat{e}\left(\gamma P, \alpha H(\varepsilon) + \sum_{i=1}^{t} \rho_{w|_{i-1}} H(w|_i)\right)}{\prod_{i=1}^{t} \hat{e}\left(\rho_{w|_{i-1}} P, \gamma H(w|_i)\right)} \\ &= \frac{\hat{e}(P, H(\varepsilon))^{\alpha\gamma} \cdot \prod_{i=1}^{t} \hat{e}\left(P, H(w|_i)\right)^{\gamma\rho_{w|_{i-1}}}}{\prod_{i=1}^{t} \hat{e}\left(P, H(w|_i)\right)^{\gamma\rho_{w|_{i-1}}}} = \hat{e}(P, H(\varepsilon))^{\gamma\alpha} \end{aligned}$$

and thus decryption recovers M.

Theorem 1 is established by the following proposition.

Proposition 1 *If $\mathcal{IG}$ satisfies the decisional BDH assumption, the above BTE scheme is secure in the sense of SN-CPA.*

Proof Let $\ell(\cdot)$ be a polynomial, and set $\ell = \ell(k)$ in what follows. Given a PPT adversary A attacking the above scheme in the sense of Definition 2, denote the probability that A succeeds by $\Pr_A[\mathsf{Succ}]$. We construct an algorithm B that attempts to solve the decisional BDH problem with respect to $\mathcal{IG}$. Relating the advantage of B to the advantage of A gives the desired result. In the description below we denote by $w|_{\bar{i}}$ the sibling of $w|_i$; namely, $w|_{\bar{i}}$ consists of the $(i-1)$-bit prefix of w followed by the negation of the i^{th} bit of w. (Thus, $w|_i$ and $w|_{\bar{i}}$ agree on their first $i-1$ bits and differ on their i^{th} bit.)

Algorithm B is given the output $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ of $\mathcal{IG}(1^k)$ and also $(P,\ Q = \alpha P,\ I_\varepsilon = \beta P,\ U_0 = \gamma P,\ d = \hat{e}(P,P)^{\mu})$; we will assume that P, Q, I_ε, U_0 are all generators of $\mathbb{G}_1$ since this occurs with all but negligible probability. The goal of B (informally) is to determine whether $\mu = \alpha\beta\gamma$. For that purpose, B simulates an instance of the encryption scheme for A as follows: B initiates a run of A, and A commits to the target node $w^* = w_1^* w_2^* \cdots w_t^*$ (with $t \le \ell$).[2] Now, for $1 \le i \le t$, B chooses χ_i, λ_i, and φ_i at random from $\mathbb{Z}_q$. If $t < \ell$, B also chooses λ_{t+1}^0, λ_{t+1}^1, and φ_{t+1} at random from $\mathbb{Z}_q$. Then B randomly chooses a hash function $H : \{0,1\}^{\le \ell} \to \mathbb{G}_1$ from the family $\mathcal{H}_\ell$ subject to the following constraints:

1. $H(\varepsilon) = I_\varepsilon$.
2. $H(w^*|_i) = \chi_i P$ for $1 \le i \le t$.
3. $H(w^*|_{\bar{i}}) = \lambda_i P - \frac{1}{\varphi_i} I_\varepsilon$ for $1 \le i \le t$.
4. If $t < \ell$, then also $H(w^*0) = \lambda_{t+1}^0 P - \frac{1}{\varphi_{t+1}} I_\varepsilon$ and $H(w^*1) = \lambda_{t+1}^1 P - \frac{1}{\varphi_{t+1}} I_\varepsilon$.

(We assume the $\{\varphi_i\}$ are invertible since this occurs with all but negligible probability.) Since there are at most $2\ell + 1$ constraints, B can efficiently choose a (random) $H \in \mathcal{H}_\ell$ subject to these constraints. Furthermore, since I_ε is uniformly distributed in $\mathbb{G}_1$ and the χ- and λ-values are all chosen independently and uniformly at random from $\mathbb{Z}_q$, this choice of H is distributed identically to H in the real experiment. B sets $PK = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, H)$ and gives PK to A.

Next, B generates secret keys for siblings of the nodes on the path from the root to w^*, as well as for the children of w^* (in case $t < \ell$). Recall that from these secret keys A can derive appropriate

[2] We assume for simplicity that $w^* \neq \varepsilon$; however, the proof may be easily adapted for that special case.

secret keys for any node w in the tree such that w is not a prefix of w^*. To generate these secret keys, B sets (for $1 \le i \le t$) $R_{w^*|_{i-1}} = \varphi_i Q$. Next, for all $1 \le i \le t$, B sets

$$S_{w^*|_{\bar{\imath}}} = \lambda_i \varphi_i Q + \sum_{j=1}^{i-1} \chi_j R_{w^*|_{j-1}} \,. \tag{1}$$

(For $i = 1$ the upper limit of the summation is less than the lower limit; by convention, we let the sum in this case be 0.) Additionally, if $t < \ell$ then B sets $R_{w^*} = \varphi_{t+1} Q$ and also (for $b \in \{0,1\}$) $S_{w^*b} = \lambda_{t+1}^b \varphi_{t+1} Q + \sum_{j=1}^{t} \chi_j R_{w^*|_{j-1}}$. Note that, having done so, B can now provide A with all relevant secret keys.

We now verify that these keys have the correct distribution. Note first that the values $R_{w^*|_0}$, …, $R_{w^*|_{t-1}}$ (and R_{w^*} when $t < \ell$) are all uniformly distributed in $\mathbb{G}_1$, independent of each other as well as PK. For $1 \le i \le t$, let $\rho_{w^*|_{i-1}} \in \mathbb{Z}_q$ be the value such that $R_{w^*|_{i-1}} = \rho_{w^*|_{i-1}} P$, and notice that $\rho_{w^*|_{i-1}} = \varphi_i \alpha$. Now, in a real execution of the experiment we would have $S_w = \alpha H(\varepsilon) + \sum_{j=1}^{|w|} \rho_{w|_{j-1}} H(w|_j)$ for any w. For $w = w^*|_{\bar{\imath}}$ this means

$$\begin{aligned} S_{w^*|_{\bar{\imath}}} &= \alpha I_\varepsilon + \left(\sum_{j=1}^{i-1} \rho_{w^*|_{j-1}} H(w^*|_j) \right) + \rho_{w^*|_{i-1}} H(w^*|_{\bar{\imath}}) \\ &= \alpha I_\varepsilon + \left(\sum_{j=1}^{i-1} \rho_{w^*|_{j-1}} \chi_j P \right) + \varphi_i \alpha (\lambda_i P - \frac{1}{\varphi_i} I_\varepsilon) \\ &= \lambda_i \varphi_i Q + \sum_{j=1}^{i-1} \chi_j R_{w^*|_{j-1}} \,, \end{aligned}$$

exactly as constructed according to Eq. (1). The analysis for the nodes w^*0 and w^*1 (in case $t < \ell$) is analogous.

After providing the appropriate secret keys, B responds to the query $\mathsf{challenge}(M_0, M_1)$ from A using the elements U_0 and d that it got as input. Specifically, B chooses a random bit b and returns

$$\begin{aligned} C &= (U_0,\ \chi_1 U_0,\ \ldots,\ \chi_t U_0,\ d \cdot M_b) \ = \ (\gamma P,\ \chi_1 \gamma P,\ \ldots,\ \chi_t \gamma P,\ \hat{e}(P,P)^\mu \cdot M_b) \\ &= (\gamma P,\ \gamma H(w^*|_1),\ \ldots,\ \gamma H(w^*),\ \hat{e}(P,P)^\mu \cdot M_b). \end{aligned}$$

Finally, if A outputs $b' = b$ then B outputs "1"; otherwise, B outputs "0".

Recalling that $Q = \alpha P$ and $H(\varepsilon) = I_\varepsilon = \beta P$, we can rewrite the last component of C as $(\hat{e}(Q, H(\varepsilon))^\gamma)^{\mu/\alpha\beta\gamma} \cdot M_b$. Thus, if $\mu = \alpha\beta\gamma$ then C is indeed a (random) valid encryption of M_b and the probability that B outputs 1 is exactly $\Pr_A[\mathsf{Succ}]$. On the other hand, when μ is random the last element of C is uniformly distributed in $\mathbb{G}_2$, independent of b, and therefore C is independent of b. In this case, then, B outputs 1 with probability 1/2. The advantage of B is therefore (negligibly close to) $|\Pr_A[\mathsf{Succ}] - 1/2|$; since the advantage of B is negligible (by assumption on $\mathcal{IG}$), the advantage of A must be negligible as well. ■

Scheme parameters. We calculate the efficiency of the above scheme as a function of the tree depth ℓ, assuming $\mathcal{H}_\ell$ is the hash family described in Section 2.2. The public key has length $\mathcal{O}(\ell)$. A secret key of a node w at level t consists of $t + 1$ elements of $\mathbb{G}_1$. (Interestingly, however, the elements $R_{w|_0}, R_{w|_1}, \ldots, R_{w|_{t-1}}$ of the secret key need not be kept secret for security to hold. This is an immediate consequence of the fact that these values are contained, anyway, in the secret keys

of the children of w.) The key-generation algorithm requires time linear in ℓ, where this complexity is due to selection of H. The key-derivation algorithm requires a constant number of operations in $\mathbb{G}_1$ and two evaluations of H; a single evaluation of H requires time $\mathcal{O}(\ell)$. Encryption for a node at level t requires t evaluations of H, $t+1$ multiplications in $\mathbb{G}_1$, one application of $\hat{e}$, and one multiplication and one exponentiation in $\mathbb{G}_2$. In the worst case, when $t = \ell$, the dominating term is the $\mathcal{O}(\ell)$ evaluations of H and thus encryption can be done in time $\mathcal{O}(\ell^2)$. For H as described in Section 2.2 this can be improved using algorithms for simultaneous polynomial evaluation at multiple points [2, Section 8.5] to yield a running time of $\mathcal{O}(\ell \log^2 \ell)$. Decryption by a node at level t requires $t+1$ evaluations of $\hat{e}$ and t multiplications/divisions in $\mathbb{G}_2$.

Construction in the random oracle model. The scheme above can be proven secure if H is replaced with a cryptographic hash function modeled as a random oracle. (A proof of security is immediate since a random oracle, in particular, acts as a $(2\ell+1)$-wise independent hash function for any polynomial ℓ.) Instantiating H in this way, and assuming that the time to evaluate H is independent of the input length, improves several of the scheme parameters: the public-key size, key-generation time, and key-derivation time are now independent of ℓ, and encryption now takes time $\mathcal{O}(\ell)$.

Once we are working in the random oracle model, the scheme may be further modified so that its security is based on the *computational* BDH assumption[3] rather than the decisional version: simply replace the component $M \cdot \hat{e}(Q, H(\varepsilon))^\gamma$ of the ciphertext by $M \oplus H'(\hat{e}(Q, H(\varepsilon))^\gamma)$, where $H' : \mathbb{G}_2 \to \{0,1\}^n$ is modeled as an independent random oracle and M is now an n-bit string.

3.2 Achieving Chosen-Ciphertext Security

We sketch how our schemes may be modified so as to achieve security in the sense of SN-CCA. In the standard model, we may apply the techniques of Sahai [37] based on earlier work of Naor and Yung [33]; namely, we may use *simulation-sound* NIZK proofs[4] [37, Definition 3.2] to achieve chosen-ciphertext security. In more detail (we assume the reader is familiar with [37]), we construct a BTE scheme secure in the sense of SN-CCA as follows: The public key consists of a randomly-generated string r for a simulation-sound NIZK proof system, as well as two independently-generated public keys PK_1, PK_2 for a BTE scheme secure in the sense of SN-CPA. The root secret key is the secret key SK_ε corresponding to PK_1, and key derivation is done in the obvious way. To encrypt message M for node w, the sender chooses random coins ω_1, ω_2, computes $C_1 = \mathsf{Enc}(PK_1, w, M; \omega_1)$, $C_2 = \mathsf{Enc}(PK_2, w, M; \omega_2)$, and then generates a simulation-sound NIZK proof of consistency π (explained in more detail below) using the string r; the output ciphertext is $C = \langle w, C_1, C_2, \pi \rangle$. The proof π guarantees that $(w, C_1, C_2, PK_1, PK_2)$ is in the $\mathcal{NP}$-language L defined by

$$\begin{aligned} L = \{&(w, C_1, C_2, PK_1, PK_2) \mid \\ &\exists M, \omega_1, \omega_2 \text{ s.t. } C_1 = \mathsf{Enc}(PK_1, w, M; \omega_1) \text{ and } C_2 = \mathsf{Enc}(PK_2, w, M; \omega_2)\}; \end{aligned}$$

that is, C_1 and C_2 are both encryptions of the same message M for the specified node w. Node w with secret key SK_w decrypts ciphertext $\langle w', C_1, C_2, \pi \rangle$ by first checking whether $w' \stackrel{?}{=} w$ and then verifying that π is a valid proof (with respect to r) of the statement $(w', C_1, C_2, PK_1, PK_2) \in L$. If so, the output is $\mathsf{Dec}(PK_1, w, SK_w, C_1)$; otherwise, the output is $\perp$. A proof that the above

[3] It is also possible to construct a scheme based on the computational BDH assumption in the standard model (by extracting hard-core bits); however, this will result in a much less efficient scheme.

[4] As in [37], we also require the proof system to satisfy the technical conditions of having *unpredictable* and *uniquely-applicable* proofs. For brevity, we do not explicitly mention this in the discussion that follows.

scheme is secure in the sense of SN-CCA exactly follows the analogous proof of [37, Thm. 4.1], and is therefore omitted.

Simulation-sound NIZK proofs (admitting efficient provers) for all of $\mathcal{NP}$ may be based on the assumption of (certified) trapdoor permutations [19, 7, 37]. We observe, additionally, that simulation-sound NIZK in the common reference string model[5] may also be based on the decisional BDH assumption. To see this, note that Sahai's construction [37] of simulation-sound NIZK requires only the existence of one-way functions (which is implied by the decisional BDH assumption) in addition to any single-theorem adaptive NIZK proof system (see Definition 2.2 of [37]). The latter, in turn, may be constructed using the "hidden-bits paradigm" set forth in [19] (see Section 4.10.2 of [22]). We observe in Appendix A that: (1) the "hidden-bits paradigm" (which is achieved using trapdoor permutations in [19]) can be implemented using what we call *publicly-verifiable trapdoor predicates*, a generalization of trapdoor permutations considered previously [18] and formally defined in Appendix A; furthermore, (2) the computational BDH assumption (and thus the decisional BDH assumption as well) gives rise to a publicly-verifiable trapdoor predicate. Finally, since the string r for the NIZK proof (included with the public key) is generated by the receiver — and not by some third party — working in the common reference string model is sufficient for our purposes.

Combining the results outlined in the preceding paragraphs yields the following:

Theorem 2 *Under the decisional BDH assumption, there exists a BTE scheme that is secure in the sense of SN-CCA.*

In the random oracle model, we can achieve a more efficient scheme secure in the sense of SN-CCA by applying, e.g., a variant of the Fujisaki-Okamoto transformation [20] (note that the Fujisaki-Okamoto transformation only applies to standard PKE and must be appropriately modified for the case of BTE). In particular: let Enc denote the encryption algorithm for a BTE scheme $\mathcal{BTE}$ secure in the sense of SN-CPA which encrypts messages at least as long as the security parameter. To simplify[6] the proof, we will assume that Enc satisfies a technical condition that we call the "unique randomness property": namely, that for any public key PK and any ciphertext C there is *at most* one set of random coins r for which there exist (w, M) satisfying

$$C = \mathsf{Enc}(PK, w, M; r).$$

(Note that for the given r, there may be multiple (w, M) satisfying the above.) It is easy to see that the BTE scheme constructed in the previous section satisfies this condition if we let r represent the random $\gamma \in \mathbb{Z}_q$ used for encryption (rather than the random bits used to generate γ). Let H and G denote independent random oracles (with appropriate ranges) which are also independent of any random oracles used by Enc. Consider then the BTE scheme in which encryption is performed as

$$\mathsf{Enc}'(PK, w, M) = \langle \mathsf{Enc}(PK, w, \sigma; H(w, \sigma, M)), G(\sigma) \oplus M \rangle$$

for randomly chosen σ of length k, the security parameter. Key generation and key derivation are done exactly as in the original BTE scheme $\mathcal{BTE}$. A node w with secret key SK_w decrypts ciphertext $\langle C_1, C_2 \rangle$ by computing $\sigma = \mathsf{Dec}(PK, w, SK_w, C_1)$ and $M = G(\sigma) \oplus C_2$. If $C_1 \stackrel{?}{=} \mathsf{Enc}(PK, w, \sigma; H(w, \sigma, M))$, the output is M; otherwise, the output is $\perp$. Security of this modified scheme is given by the following theorem, whose proof appears in Appendix B:

[5] In the common reference string model — as distinguished from the common random string model — the string r may be chosen from an arbitrary, poly-time computable distribution (and not necessarily a uniform one); furthermore, the coins used to generate r are kept secret.

[6] A proof does not seem to require this assumption, but making the assumption allows us to avoid having to deal with some annoying technicalities. See footnote 7 in Appendix B.

Theorem 3 *If $\mathcal{BTE}$ is secure in the sense of SN-CPA and satisfies the unique randomness property, then the above construction yields a BTE scheme secure in the sense of SN-CCA in the random oracle model.*

4 Forward-Secure Public-Key Encryption

Here, we provide a definition of security for forward-secure public-key encryption and mention two "trivial" forward-secure schemes whose complexity is linear in the total number of time periods. As our main result of this section, we then describe a construction of a forward-secure encryption scheme all of whose parameters grow at most *poly-logarithmically* with the total number of time periods. This construction builds on the BTE primitive discussed in the previous section.

4.1 Definitions

We first provide a syntactic definition of key-evolving public-key encryption schemes, and then define what it means for such a scheme to achieve forward security. The former is a straightforward adaptation of the notion of key-evolving signature schemes [5]; the latter, however, is new.

Definition 4 A (public-key) *key-evolving encryption* (ke-PKE) *scheme* is a 4-tuple of PPT algorithms $(\mathsf{Gen}, \mathsf{Upd}, \mathsf{Enc}, \mathsf{Dec})$ such that:

- The *key generation algorithm* Gen takes as input a security parameter 1^k and the total number of time periods N. It returns a public key PK and an initial secret key SK_0. (We assume N is implicit in PK.)
- The *key update algorithm* Upd takes as input PK, an index $i \in [0, N-1)$ of the current time period, and the associated secret key SK_i. It returns the secret key SK_{i+1} for the following time period.
- The *encryption algorithm* Enc takes as input PK, an index $i \in [0, N)$ of a time period, and a message M. It returns a ciphertext C.
- The *decryption algorithm* Dec takes as input PK, an index $i \in [0, N)$ of the current time period, the associated secret key SK_i, and a ciphertext C. It returns a message M or $\perp$.

We make the obvious correctness requirement: namely, for any (PK, SK_0) output by $\mathsf{Gen}(1^k, N)$, any index $i \in [0, N)$ and secret key SK_i correctly generated for this time period, and any message M, we have $M = \mathsf{Dec}(PK, i, SK_i, \mathsf{Enc}(PK, i, M))$. $\diamondsuit$

Our definitions of forward-secure public-key encryption generalize the standard notions of security for public-key encryption, similar to the way in which the definitions of [5] generalize the standard notion of security for signature schemes.

Definition 5 A ke-PKE scheme is *forward-secure against chosen-plaintext attacks* (secure in the sense of fs-CPA) if for all polynomials $N(\cdot)$, the advantage of any PPT adversary in the following game is negligible in the security parameter k (in the following, let $N = N(k)$):

Setup: $\mathsf{Gen}(1^k, N)$ outputs (PK, SK_0). The adversary is given PK.

Attack: The adversary issues one $\mathsf{breakin}(i)$ query and one $\mathsf{challenge}(j, M_0, M_1)$ query, in either order, subject to $0 \leq j < i < N$. These queries are answered as follows:

- On query $\mathsf{breakin}(i)$, key SK_i is computed via repeated application of Upd in the obvious way. This key is then given to the adversary.
- On query $\mathsf{challenge}(j, M_0, M_1)$, a random bit b is selected and the adversary is given $C^* = \mathsf{Enc}(PK, j, M_b)$.

Guess: The adversary outputs a guess $b' \in \{0,1\}$; it succeeds if $b' = b$. The adversary's *advantage* is the absolute value of the difference between its success probability and 1/2. ♢

We give an analogous definition incorporating chosen-ciphertext attacks by the adversary.

Definition 6 A ke-PKE scheme is *forward-secure against chosen-ciphertext attacks* (secure in the sense of fs-CCA) if for all polynomials $N(\cdot)$, the advantage of any PPT adversary in the following game is negligible in the security parameter k (again, let $N = N(k)$):

Setup: $\mathsf{Gen}(1^k, N)$ outputs (PK, SK_0). The adversary is given PK.

Attack: The adversary issues one $\mathsf{breakin}(i)$ query, one $\mathsf{challenge}(j, M_0, M_1)$ query, and multiple $\mathsf{Dec}^*(k, C)$ queries, in any order, subject to $0 \le j < i < N$ and $k \in [0, N)$. These queries are answered as follows:

- The breakin and challenge queries are answered as in Definition 5.
- On query $\mathsf{Dec}^*(k, C)$, the appropriate key SK_k is first derived via repeated application of Upd in the obvious way. The adversary is then given the output $\mathsf{Dec}(PK, k, SK_k, C)$. If the adversary has already received response C^* from query $\mathsf{challenge}(j, M_0, M_1)$, then query $\mathsf{Dec}^*(j, C^*)$ is disallowed (but queries $\mathsf{Dec}^*(k, C^*)$ with $k \neq j$, and $\mathsf{Dec}^*(j, C)$ with $C \neq C^*$, *are* allowed).

Guess: The adversary outputs a guess $b' \in \{0,1\}$; it succeeds if $b' = b$. The adversary's *advantage* is the absolute value of the difference between its success probability and 1/2. ♢

The discussion in Remark 1 applies here as well in case the key-update algorithm is randomized. Actually, things are slightly easier here: since N is polynomial in k, we may as well assume that all secret keys for all time periods are generated at the outset of the experiment, and then used (as needed) to answer the oracle queries of A.

Remark 2 (On the Order of the Breakin/Challenge Queries). The definitions above allow the adversary to make the breakin and the challenge queries in either order. Without loss of generality, however, we may assume the adversary makes the breakin query first. (Specifically, given an adversary A that queries $\mathsf{challenge}(j, M_0, M_1)$ before its breakin query, it is easy to construct an adversary B that queries $\mathsf{breakin}(j+1)$ followed by this same challenge query and can then answer any subsequent breakin query of A; this B will achieve the same advantage as A.)

Interestingly, requiring the adversary to make the challenge query first seems to result in slightly weaker concrete security. Specifically, transforming an adversary that first makes the breakin query into an adversary that first makes the challenge query results in a factor of N degradation in the advantage due to the need to guess the location of the eventual challenge query in advance. Since N is polynomial in k, this reduction in security is tolerable. Still, it is better to avoid it.

4.2 Forward-Secure PKE Schemes with Linear Complexity

For completeness, we discuss some simple approaches to forward-secure PKE yielding schemes with linear complexity in at least some parameters. One trivial solution is to generate N independent public-/private-key pairs $\{(sk_i, pk_i)\}$ for any standard PKE scheme and to set $PK = (pk_0, \ldots, pk_{N-1})$. In this scheme, the key SK_i for time period i will simply consist of $(sk_i, \ldots, sk_{N-1})$. Algorithms for encryption, decryption, and key update are immediate. The drawback of this trivial solution is an N-fold increase in the sizes of the public and secret keys, as well as in the key-generation time. Anderson [3] noted that a slightly improved solution can be built using any *identity-based* encryption scheme. Here, the public key is the "master public key" of the identity-based scheme, and SK_i is the secret key corresponding to the "identity" i (the scheme is otherwise identical to the above). This solution achieves $\mathcal{O}(1)$ public key size, but still has $\mathcal{O}(N)$ secret-key size and key-generation time.

One can improve upon this last solution somewhat: instead of a large secret key, the user may store a large *non-secret* file containing one record per period. The record for period i contains the secret key SK_i encrypted with respect to the public key and time period $i-1$. At the beginning of period i, the user obtains record i, uses its current key SK_{i-1} to recover SK_i, and then erases SK_{i-1}. This solution achieves essentially the same efficiency as the forward-secure signatures of Krawczyk [30] and in particular requires $\mathcal{O}(N)$ non-secret storage and key-generation time.

4.3 A Construction with Poly-Logarithmic Complexity in All Parameters

We now construct an encryption scheme secure in the sense of fs-CPA (resp., fs-CCA) from any BTE scheme secure in the sense of SN-CPA (resp., SN-CCA). Our construction is straightforward and is easily seen to be secure given the machinery we have developed for BTE schemes in Section 3.

At a high level, the construction proceeds as follows: To obtain a forward-secure scheme with $N = 2^{\ell+1} - 1$ time periods (labeled 0 through $N-1$), we use a BTE of depth ℓ and associate the time periods with all nodes of the tree according to a pre-order traversal. Namely, letting w^i denote the node associated with period i, we have:

- $w^0 = \varepsilon$ (i.e., the root of the tree).
- If w^i is an *internal node* then $w^{i+1} = w^i 0$.
- If w^i is a *leaf node* and $i < N-1$ then $w^{i+1} = w'1$, where w' is the longest string such that $w'0$ is a prefix of w^i.

The public key will simply be the public key for the BTE scheme; the secret key for period i will consist of the secret key (in the underlying BTE scheme) for node w^i as well as the secret keys for all right siblings of the nodes on the path from the root to w^i. To encrypt a message at time period i, the message is simply encrypted for node w^i using the BTE scheme; decryption is done in the obvious way using the secret key for node w^i (which is stored as part of the secret key for period i). Finally, the secret key is updated at the end of period i in the following manner: if w^i is an internal node, then the secret keys for w^{i+1} and its sibling (i.e., the two children of w^i) are derived as in the underlying BTE scheme; otherwise, the secret key for node w^{i+1} is already stored as part of the secret key. In either case, the key for node w^i is then deleted. Note that this maintains the property that SK_{i+1} contains the secret key for w^{i+1} as well as the secret keys for all right siblings of the nodes on the path from the root to w^{i+1}. Also, only $\mathcal{O}(\ell)$ secret keys of the underlying BTE scheme are stored as part of the secret key of the forward-secure scheme at any point in time.

Our method of associating time periods with nodes of a binary tree is reminiscent of previous tree-based forward-secure signature schemes [5, 1, 31]. However, we associate time periods with *all* nodes of a binary tree rather than with the leaves only (as was done in prior work); this results in an efficiency improvement from $\mathcal{O}(\log N)$ to $\mathcal{O}(1)$ in the key-generation and (worst-case) key-update times. We remark that our tree-traversal method can also be applied to the signature schemes of [5, 1, 31] with similar efficiency gains for the worst-case complexity of these algorithms.

More formally, given a BTE scheme (Gen, Der, Enc, Dec), we may construct a ke-PKE scheme (Gen', Upd, Enc', Dec') as follows:

- Algorithm $\mathsf{Gen}'(1^k, N)$ runs $\mathsf{Gen}(1^k, 1^\ell)$, where ℓ is the smallest integer satisfying $N \leq 2^{\ell+1}-1$, and obtains PK, SK_ε. It then outputs $PK' = (PK, N)$, and $SK'_0 = SK_\varepsilon$.
- Algorithm $\mathsf{Upd}(PK, i, SK'_i)$ has SK'_i organized as a stack of node keys, with the secret key SK_{w^i} on top. We first pop this key off the stack. If w^i is a leaf node, the next key on top of the stack is $SK_{w^{i+1}}$ and we are done. If w^i is an internal node, compute $(SK_{w^i 0}, SK_{w^i 1}) \leftarrow \mathsf{Der}(PK, w^i, SK_{w^i})$ and push $SK_{w^i 1}$ and then $SK_{w^i 0}$ onto the stack. The new key on top of the stack is $SK_{w^i 0}$ (and indeed $w^{i+1} = w^i 0$). In either case, node key SK_{w^i} is then erased and the new stack of node keys is returned.
- Algorithm $\mathsf{Enc}'(PK', i, M)$ runs $\mathsf{Enc}(PK, w^i, M)$. Note that w^i is publicly computable (in $\mathcal{O}(\log N)$ time) given i and N.
- Algorithm $\mathsf{Dec}'(PK', i, SK'_i, M)$ runs $\mathsf{Dec}(PK, w^i, SK_{w^i}, M)$, where SK_{w^i} is the node key on top of the stack of keys stored as part of SK'_i.

Theorem 4 *If BTE scheme* ($\mathsf{Gen}, \mathsf{Der}, \mathsf{Enc}, \mathsf{Dec}$) *is secure in the sense of SN-CPA (resp., SN-CCA) then ke-PKE scheme* ($\mathsf{Gen}', \mathsf{Upd}, \mathsf{Enc}', \mathsf{Dec}'$) *is secure in the sense of fs-CPA (resp., fs-CCA).*

Proof The proof proceeds via a straightforward reduction. Assume we have an adversary A' with advantage $\mathsf{Adv}(k)$ in an fs-CPA (resp., fs-CCA) attack against ($\mathsf{Gen}', \mathsf{Upd}, \mathsf{Enc}', \mathsf{Dec}'$). We construct an adversary A with advantage $\mathsf{Adv}(k)/N(k)$ in the corresponding attack against the underlying BTE scheme ($\mathsf{Gen}, \mathsf{Der}, \mathsf{Enc}, \mathsf{Dec}$). Since $N = N(k)$ is polynomial in the security parameter k, the theorem follows. We now define adversary A:

1. A chooses uniformly at random a time period $i^* \in [0, N)$ and outputs w^{i^*}. Next, A obtains the public key PK and the appropriate secret keys for the BTE scheme.
2. A runs A' with public key (PK, N).
3. When A' queries $\mathsf{breakin}(j)$ (recall from Remark 2 that without loss of generality A' makes its $\mathsf{breakin}$ query before its $\mathsf{challenge}$ query), if $j \leq i^*$ then A outputs a random bit and halts. Otherwise, A computes the appropriate secret key SK'_j and gives this to A'. (Observe that A can efficiently compute SK'_j for $j > i^*$ from the secret keys it has been given.)
4. When A' queries $\mathsf{challenge}(i, M_0, M_1)$, if $i \neq i^*$ then A outputs a random bit and halts. Otherwise, A obtains $C \leftarrow \mathsf{challenge}(M_0, M_1)$ and gives ciphertext C to A'.
5. If decryption queries are allowed, note that A can respond to queries $\mathsf{Dec}'^*(k, C)$ of A' by simply querying $\mathsf{Dec}^*(w^k, C)$ and returning the result to A'.
6. When A' outputs b', A outputs b' and halts.

It is straightforward to see that when $i^* = i$ the copy of A' running within A has exactly the same view as in a real fs-CPA (resp., fs-CCA) interaction. Since A guesses $i^* = i$ with probability $1/N$, we have that A correctly predicts the bit b with advantage $\mathsf{Adv}(k)/N$. ■

Scheme parameters. Each of the four operations of the FSE scheme (key generation, key update, encryption, and decryption) requires at most one corresponding operation of an underlying BTE scheme of depth $\ell = \mathcal{O}(\log N)$. The secret key of the FSE scheme at any time period consists of at most $\mathcal{O}(\log N)$ node secret keys of the underlying BTE scheme. Since node secret keys in the BTE scheme constructed in Section 3.1 are of size at most $\mathcal{O}(\log N)$, this immediately implies an FSE scheme in which secret keys have size $\mathcal{O}(\log^2 N)$. For the specific construction of a BTE scheme given in Section 3.1, however, we may notice that for any node w at depth $|w| = t$ all elements of the node secret key except for $R_{w|_{t-1}}$ and S_w already appear in the secret key of the parent of w. Thus, when using this BTE scheme to construct an FSE scheme, secret keys for the FSE scheme can in fact be stored using only $\mathcal{O}(\log N)$ space. This justifies the claims given in Table 1 (for schemes achieving security in the sense of fs-CPA), and yields the following corollary:

Corollary 1 *Under the decisional BDH assumption, there exists a ke-PKE scheme that is secure in the sense of fs-CPA. Furthermore, all parameters of this scheme are poly-logarithmic in the total number of time periods.*

Supporting an unbounded number of time periods. In our description above, we have assumed that the number of time periods N is known at the time of key generation. However, it is easy to modify our scheme to support an "unbounded" (i.e., arbitrary polynomial) number of time periods by using a BTE scheme with depth $\ell = \omega(\log k)$. Following the techniques of [31], we can further improve this scheme so that its efficiency depends only poly-logarithmically on the number of time periods *elapsed thus far* (note that a simple pre-order traversal using a tree of depth $\omega(\log k)$ results in a scheme with super-logarithmic dependence on N for any $N = \mathsf{poly}(k)$).

5 Hierarchical Identity-Based Encryption

Here we show how one can construct a full-blown hierarchical identity-based encryption (HIBE) scheme from any BTE scheme. (As noted in the introduction, the security we obtain for the resulting identity-based scheme is slightly weaker than the notion of security considered in earlier work on identity-based encryption [9, 21].)

We begin by providing a syntactic definition of HIBE essentially following [26, 21]. We then introduce the notion of "selective identity" security for HIBE, and show how to transform any secure BTE scheme into a secure HIBE scheme.

5.1 Definitions

In all the definitions below, an *ID-vector* v is a vector of strings, i.e., $v \in (\{0,1\}^*)^*$. The empty vector is denoted by $()$. If $v = (v_1, \ldots, v_\ell)$ is an ID-vector and $v_{\ell+1}$ is any string, then by $v.v_{\ell+1}$ we mean the ID-vector $(v_1, \ldots, v_\ell, v_{\ell+1})$. For two ID-vectors $u = (u_1, \ldots, u_{\ell_1})$ and $v = (v_1, \ldots, v_{\ell_2})$, we say that u is a prefix of v if $\ell_1 \leq \ell_2$ and $u_i = v_i$ for $i \leq \ell_1$.

Definition 7 A hierarchical identity-based encryption (HIBE) scheme is a 4-tuple of PPT algorithms $(\mathsf{Gen}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$ such that:

- The *key-generation algorithm* Gen takes as input a security parameter 1^k and a value 1^ℓ for the depth of the tree. It returns a master public key PK and a root secret key $SK_{()}$. We assume that 1^k and 1^ℓ are implicit in PK.

- The *key-extraction algorithm* Ext takes the public key PK, an ID-vector $v \in (\{0,1\}^*)^{<\ell}$ and its associated secret key SK_v, and a string r. It returns the secret key $SK_{v.r}$ associated with the ID-vector $v.r$.

- The *encryption algorithm* Enc takes a public key PK, an ID-vector $v \in (\{0,1\}^*)^{\leq\ell}$, and a message M. It returns a ciphertext C.

- The *decryption algorithm* Dec takes as input a public key PK, an ID-vector $v \in (\{0,1\}^*)^{\leq\ell}$ and its associated secret key SK_v, and a ciphertext C. It returns a message M or symbol $\perp$.

We make the natural correctness requirement: namely, for any $(PK, SK_{()})$ output by $\mathsf{Gen}(1^k, 1^\ell)$, any ID-vector $v \in (\{0,1\}^*)^{\leq\ell}$ and secret key SK_v correctly generated for this ID-vector, and any message M, we have $M = \mathsf{Dec}(PK, v, SK_v, \mathsf{Enc}(PK, v, M))$. $\diamondsuit$

The notion of "selective identity" security we present is a relaxation of the notion of security for IBE/HIBE schemes considered previously, and requires that the attacker commit to a "target" ID-vector (or, in the case of IBE, a "target" identity) before it sees the public key. (Previous definitions allow the adversary to choose the target ID-vector/identity adaptively, as a function of the public key as well as any secret keys it obtains.) Other than this, our definition follows that given in [21]. We provide a definition for the case of chosen-plaintext security; an analogous definition of security against selective-identity, chosen-ciphertext attacks ($\mathsf{SI\text{-}CCA}$) is the obvious extension of this, and is omitted.

Definition 8 A HIBE scheme is *secure against selective-identity, chosen-plaintext attacks* ($\mathsf{SI\text{-}CPA}$) if for all polynomials $\ell(\cdot)$, the advantage of any PPT adversary A in the following game is negligible in the security parameter k (we set $\ell = \ell(k)$ in what follows):

1. The adversary $A(1^k, 1^\ell)$ outputs an ID-vector $v^* \in (\{0,1\}^*)^{\leq\ell}$.
2. Algorithm $\mathsf{Gen}(1^k, 1^\ell)$ outputs $(PK, SK_{()})$. The adversary is given PK.
3. The adversary may adaptively ask for the secret key(s) corresponding to any ID-vector(s) v, as long as v is not a prefix of the target ID-vector v^*. The adversary is given the secret key SK_v correctly generated for v using the Ext algorithm.
4. The adversary generates a request $\mathsf{challenge}(M_0, M_1)$ with $|M_0| = |M_1|$. A random bit b is selected and the adversary is given $C^* = \mathsf{Enc}(PK, v^*, M_b)$.
5. The adversary can keep asking for secret keys as above, even after seeing C^*.

At the end of the game the adversary outputs $b' \in \{0,1\}$; it succeeds if $b' = b$. The adversary's *advantage* is the absolute value of the difference between its success probability and $1/2$. $\diamondsuit$

The discussion in Remark 1 applies here as well in case the key-extraction algorithm is randomized.

5.2 From BTE to HIBE

We now show the transformation from a BTE scheme to a HIBE scheme. In the transformation, we will use universal one-way hashing [32] to map an ID-vector with a bounded number of entries to a bounded-length string by applying the hash function separately to each entry in the vector and then concatenating the results. We thus obtain a string whose length depends only on the number of entries in the input ID-vector (and not the length of these entries).

Universal one-way hashing. A universal one-way hash function (UOWHF) [32] consists of two algorithms: a seed-generation algorithm sGen that (given the security parameter k in unary) outputs a seed s, and a hashing algorithm Hash that given a seed s and an input string of some polynomial length, produces a k-bit output string. A *collision* for a seed s is a pair of distinct inputs x, x' such that $\mathsf{Hash}(s,x) = \mathsf{Hash}(s,x')$. Security of a UOWHF $(\mathsf{sGen}, \mathsf{Hash})$ requires that no adversary can find a collision involving an input string x chosen *before* selection of s; formally, for all PPT A the following is negligible:

$$\Pr[x \leftarrow A(1^k); s \leftarrow \mathsf{sGen}(1^k); x' \leftarrow A(1^k, s, x) : x' \neq x \wedge \mathsf{Hash}(s,x') = \mathsf{Hash}(s,x)].$$

We remark that a collision-resistant hash function [15] is also a UOWHF; however, constructions of UOWHFs are known based on the minimal assumption of one-way functions [36].

It will be convenient to define some notation for the entry-wise application of a hash function to an ID-vector. If s is a seed output by $\mathsf{sGen}(1^k)$ and $v = (v_1, \ldots, v_\ell)$ is an ID-vector, then we let $w = \mathsf{Hash}(s,v)$ refer to the string $\mathsf{Hash}(s,v_1)|\cdots|\mathsf{Hash}(s,v_\ell)$. Note that if $v \in (\{0,1\}^*)^t$ then $w \in \{0,1\}^{kt}$.

The construction. Our construction proceeds by identifying the ID-vector $v \in (\{0,1\}^*)^{\leq \ell}$ with the node $w = H(s,v)$ in a binary tree of depth $k\ell$; then, to encrypt a message destined for user v, the message is encrypted for this node w using an underlying BTE scheme. In more detail, given a UOWHF $(\mathsf{sGen}, \mathsf{Hash})$ and a BTE scheme $(\mathsf{Gen}, \mathsf{Der}, \mathsf{Enc}, \mathsf{Dec})$, we may construct a HIBE scheme $(\mathsf{Gen}', \mathsf{Ext}, \mathsf{Enc}', \mathsf{Dec}')$ as follows:

- Algorithm $\mathsf{Gen}'(1^k, 1^\ell)$ runs $\mathsf{Gen}(1^k, 1^{k\ell})$ and obtains (PK, SK_ε). It also runs $\mathsf{sGen}(1^k)$ and obtains a seed s. The public key of the HIBE is $PK' = (s, PK)$ and the root secret key is $SK'_{()} = SK_\varepsilon$.
- Algorithm $\mathsf{Ext}(PK', v, SK'_v, r)$ sets $w = \mathsf{Hash}(s,v)$ and $w' = \mathsf{Hash}(s,r)$ (with $|w'| = k$). For $i = 1, \ldots, k$, algorithm Ext uses algorithm Der to derive the BTE secret key $SK_{w(w'|_i)}$ from the BTE secret key $SK_{w(w'|_{i-1})}$. (Recall that the given secret key SK'_v is nothing more than the secret key SK_w for the BTE scheme.) The HIBE secret key is then set to $SK'_{v.r} = SK_{ww'}$.
- Algorithm $\mathsf{Enc}'(PK', v, M)$ runs $\mathsf{Enc}(PK, w, M)$, where $w = \mathsf{Hash}(s,v)$.
- Algorithm $\mathsf{Dec}'(PK', v, SK'_v, M)$ runs $\mathsf{Dec}(PK, w, SK_w, M)$, where $w = \mathsf{Hash}(s,v)$.

Theorem 5 *If* $(\mathsf{sGen}, \mathsf{Hash})$ *is a UOWHF and* $(\mathsf{Gen}, \mathsf{Der}, \mathsf{Enc}, \mathsf{Dec})$ *is a BTE scheme secure in the sense of SN-CPA (resp., SN-CCA), then* $(\mathsf{Gen}', \mathsf{Ext}, \mathsf{Enc}', \mathsf{Dec}')$ *is a HIBE scheme secure in the sense of SI-CPA (resp., SI-CCA).*

Proof The proof is immediate. Given an adversary A that attacks the HIBE scheme (in either the CPA or CCA scenario), we build an adversary B that attacks the underlying BTE scheme (in

the same scenario). The adversary B implements for A an HIBE scheme exactly as above, choosing the seed s for the hash function according to $\mathsf{sGen}(1^k)$.

When A commits to its target ID-vector v^*, the adversary B commits to its target node $w^* = \mathsf{Hash}(s, v^*)$. Then B uses its own queries to the BTE scheme to answer all of the queries that A makes to the HIBE scheme, with only two possible exceptions. One exception occurs in case A asks for a secret key SK'_v, corresponding to ID-vector v, such that v is not a prefix of the target ID-vector v^* but $w = \mathsf{Hash}(s, v)$ is a prefix of the target node $w^* = \mathsf{Hash}(s, v^*)$. The other exception (that can only occur in the CCA scenario) occurs in case A asks a decryption query $\mathsf{Dec}^*(v, C^*)$ such that $v \neq v^*$ but $\mathsf{Hash}(s, v) = \mathsf{Hash}(s, v^*)$. It is easy to see that either of these cases yields a collision in the hash function involving an entry in the ID-vector v^*. Since A commits to the target ID-vector v^* before obtaining the seed s (which is included as part of the public key), a straightforward hybrid argument shows that the probability of such a collision occurring is negligible. Thus, B's advantage is only negligibly smaller than the advantage of A. ■

Since a universal one-way hash function may be constructed from any one-way function [36] (and thus, in particular, from any BTE scheme), we obtain the following result:

Theorem 6 *Assuming the existence of a BTE scheme secure in the sense of* SN-CPA *(resp.,* SN-CCA*), there exists a HIBE scheme secure in the sense of* SI-CPA *(resp.,* SI-CCA*).*

Acknowledgments

The third author is very grateful to Craig Gentry for helpful discussions regarding [21] and for providing a preliminary version of that work. We also thank the anonymous referees for their helpful feedback.

References

[1] M. Abdalla and L. Reyzin. A New Forward-Secure Digital Signature Scheme. *Advances in Cryptology — Asiacrypt 2000*, LNCS vol. 1976, Springer-Verlag, pp. 116–129, 2000.

[2] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[3] R. Anderson. Two Remarks on Public Key Cryptology. Invited Lecture, *ACM CCCS '97.* Available at `http://www.cl.cam.ac.uk/ftp/users/rja14/forwardsecure.pdf`.

[4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. *Advances in Cryptology — Crypto '98*, LNCS vol. 1462, Springer-Verlag, pp. 26–45, 1998.

[5] M. Bellare and S. K. Miner. A Forward-Secure Digital Signature Scheme. *Advances in Cryptology — Crypto '99*, LNCS vol. 1666, Springer-Verlag, pp. 431–448, 1999.

[6] M. Bellare and B. Yee. Forward Security in Private-Key Cryptography. *RSA Cryptographers' Track — CT-RSA 2003*, LNCS vol. 2612, Springer-Verlag, pp. 1–18, 2003.

[7] M. Bellare and M. Yung. Certifying Permutations: Non-Interactive Zero-Knowledge Based on any Trapdoor Permutation. *J. Cryptology* 9(3): 149–166 (1996).

[8] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical Identity Based Encryption with Constant Size Ciphertext. *Advances in Cryptology — Eurocrypt 2005*, LNCS vol. 3494, Springer-Verlag, pp. 440–456, 2005.

[9] D. Boneh and M. Franklin. Identity Based Encryption from the Weil Pairing. *SIAM J. Computing* 32(3): 586–615 (2003).

[10] D. Boneh and J. Katz. Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity-Based Encryption. *RSA Cryptographers' Track — CT-RSA 2005*, LNCS vol. 3376, Springer-Verlag, pp. 87–103, 2005.

[11] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited. *J. ACM* 51(4): 557–594 (2004).

[12] R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. *Advances in Cryptology — Eurocrypt 2003*, LNCS vol. 2656, Springer-Verlag, pp. 255–271, 2003.

[13] R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. *Advances in Cryptology — Eurocrypt 2004*, LNCS vol. 3027, Springer-Verlag, pp. 207–222, 2004.

[14] R. Canetti, S. Halevi, and J. Katz. Adaptively-Secure, Non-Interactive Public-Key Encryption. *2nd Theory of Cryptography Conference (TCC)*, LNCS vol. 3378, Springer-Verlag, pp. 150–168, 2005. Full version available at `http://eprint.iacr.org/2004/317`.

[15] I. Damgård. Collision Free Hash Functions and Public-Key Signature Schemes. *Advances in Cryptology — Eurocrypt '87*, LNCS vol. 304, Springer-Verlag, pp. 203–216, 1988.

[16] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. *Advances in Cryptology — Crypto '89*, LNCS vol. 435, Springer-Verlag, pp. 307–315, 1990.

[17] W. Diffie, P. C. Van-Oorschot, and M. J. Weiner. Authentication and Authenticated Key Exchanges. *Designs, Codes, and Cryptography* 2(2): 107–125 (1992).

[18] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong Key-Insulated Signature Schemes. *Public-Key Cryptography — PKC 2003*, LNCS vol. 2567, Springer-Verlag, pp. 130–144, 2003.

[19] U. Feige, D. Lapidot, and A. Shamir. Multiple Non-Interactive Zero-Knowledge Proofs Under General Assumptions. *SIAM J. Computing* 29(1): 1–28 (1999).

[20] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *Advances in Cryptology — Crypto '99*, LNCS vol. 1666, Springer-Verlag, pp. 537–554, 1999.

[21] C. Gentry and A. Silverberg. Hierarchical Identity-Based Cryptography. *Advances in Cryptology — Asiacrypt 2002*, LNCS vol. 2501, Springer-Verlag, pp. 548–566, 2002.

[22] O. Goldreich. *Foundations of Cryptography, vol. 1: Basic Tools*. Cambridge University Press, 2001.

[23] O. Goldreich. *Foundation of Cryptography, vol. 2: Basic Applications*. Cambridge University Press, 2004.

[24] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Computing* 17(2): 281–308 (1988).

[25] C.G. Günther. An Identity-Based Key-Exchange Protocol. *Advances in Cryptology — Eurocrypt '89*, LNCS vol. 434, Springer-Verlag, pp. 29–37, 1990.

[26] J. Horwitz and B. Lynn. Toward Hierarchical Identity-Based Encryption. *Advances in Cryptology — Eurocrypt 2002*, LNCS vol. 2332, Springer-Verlag, pp. 466–481, 2002.

[27] G. Itkis and L. Reyzin. Forward-Secure Signatures with Optimal Signing and Verifying. *Advances in Cryptology — Crypto 2001*, LNCS vol. 2139, Springer-Verlag, pp. 499–514, 2001.

[28] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in Cryptographic Groups. Manuscript, January 2001. Available at `http://eprint.iacr.org/2001/003/`.

[29] A. Kozlov and L. Reyzin. Forward-Secure Signatures with Fast Key Update. *Security in Communication Networks*, LNCS vol. 2576, Springer-Verlag, pp. 247–262, 2002.

[30] H. Krawczyk. Simple Forward-Secure Signatures From any Signature Scheme. *10th ACM Conference on Computer and Communications Security*, ACM, pp. 108–115, 2000.

[31] T. Malkin, D. Micciancio, and S. K. Miner. Efficient Generic Forward-Secure Signatures with an Unbounded Number of Time Periods. *Advances in Cryptology — Eurocrypt 2002*, LNCS vol. 2332, Springer-Verlag, pp. 400–417, 2002.

[32] M. Naor and M. Yung. Universal One-Way Hash Functions and Their Cryptographic Applications. *21st ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 33–43, 1989.

[33] M. Naor and M. Yung, Public Key Cryptosystems Provably Secure Against Chosen Ciphertext Attacks. *22nd ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 427–437, 1990.

[34] R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. *10th ACM Symposium on Principles of Distributed Computing (PODC)*, ACM, pp. 51–59, 1991.

[35] C. Rackoff and D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext attack. *Advances in Cryptology — Crypto '91*, LNCS vol. 576, Springer-Verlag, pp. 433–444, 1992.

[36] J. Rompel. One-Way Functions are Necessary and Sufficient for Secure Signatures. *22nd ACM Symposium on Theory of Computing (STOC)*, ACM, pp. 387–394, 1990.

[37] A. Sahai. Non-Malleable Non-Interactive Zero-Knowledge and Adaptive Chosen-Ciphertext Security. *40th IEEE Symposium on Foundations of Computer Science (FOCS)*, IEEE, pp. 543–553, 1999.

[38] A. Shamir. How to Share a Secret. *Comm. of the ACM* 22(11): 612–613 (1979).

A Basing NIZK on the (Computational) BDH Assumption

In this section we show two results culminating in a construction of an NIZK proof system (for all of $\mathcal{NP}$) in the common reference string model (see footnote 5) based on the computational BDH assumption. Recall that this can then be used to achieve chosen-ciphertext security for our BTE scheme in the standard model.

First, we formally define a new primitive which we call a *publicly-verifiable trapdoor predicate* (first suggested in [18]), and show that the computational BDH assumption can be used to construct such a primitive. This new primitive may be viewed as a generalization of trapdoor permutations, and indeed we argue that the construction of an NIZK proof system based on trapdoor permutations given by Feige-Lapidot-Shamir [19] (in the common *random* string model) can in fact be based on publicly-verifiable trapdoor permutations in the common *reference* string model. Finally, we note that the publicly-verifiable trapdoor predicate which arises naturally from the computational BDH assumption is sufficient for NIZK *in the context of CCA2-secure encryption* (see discussion below).

We begin with a definition of publicly-verifiable trapdoor predicates. As noted above, these may be viewed as generalizing the notion of trapdoor permutations. Somewhat informally, we replace the requirements that (1) the domain of the permutation π is efficiently sampleable and that (2) π is efficiently computable, by the (weaker) requirements that (1) it is possible to efficiently sample pairs $(x, \pi(x))$ uniformly at random and that (2) given a pair (x, y) it is possible to efficiently determine whether or not $y = \pi(x)$. The formal definition we give here is patterned after the definition of trapdoor permutations [22, Definition 2.4.5]. Below we let $\bar{I} \subseteq \{0,1\}^*$ be an index set, and corresponding to each index $i \in \bar{I}$ we associate a domain D_i and a predicate $f_i : D_i \times D_i \to \{0,1\}$. (Informally, f_i indicates whether or not the pair (x, y) is of the appropriate form.)

Definition 9 Let $\mathcal{F} = \{f_i : i \in \bar{I}\}$ be a collection of functions $f_i : D_i \times D_i \to \{0,1\}$ such that for all $i \in \bar{I}$ and $y \in D_i$, there is a unique x for which $f_i(x, y) = 1$. Collection $\mathcal{F}$ is a *publicly-verifiable trapdoor predicate* if there exist four PPT algorithms I, D, F, F^{-1} such that:

- *Index and trapdoor selection*: For all k we have $I(1^k) \in \bar{I} \times \{0,1\}^*$.
- *Uniform sampling of valid predicates*: For all $i \in \bar{I}$ we have
  - If $(x, y) \leftarrow D(i)$ then $f_i(x, y) = 1$.
  - The distribution $\{(x, y) \leftarrow D(i) : y\}$ is exactly the uniform distribution over D_i.
- *Efficient predicate evaluation*: For all $i \in \bar{I}$ and all $(x, y) \in D_i \times D_i$, $F(i, x, y) = f_i(x, y)$.
- *Hard to find a valid "match"*: For all PPT algorithms A the following is negligible in k:

$$\Pr[(i, td) \leftarrow I(1^k); (x, y) \leftarrow D(i) : A(1^k, i, y) = x].$$

- *Easy to find a valid "match" with the trapdoor*: For all k, any pair (i, td) output by $I(1^k)$, and any $y \in D_i$ we have $f_i(F^{-1}(td, y), y) = 1$.

$\diamondsuit$

It is not hard to see that a BDH parameter generator $\mathcal{IG}$ satisfying the computational BDH assumption (see Section 2.1) gives rise to a publicly-verifiable trapdoor predicate. Informally, this predicate arises because the computational Diffie-Hellman problem in $\mathbb{G}_1$ is "hard" while the decisional Diffie-Hellman problem in $\mathbb{G}_1$ is "easy" (see [28]). Specifically, having the index include

the output of $\mathcal{IG}$ and a pair of random elements $P, Q \in \mathbb{G}_1$ (and letting the output of $\mathcal{IG}$ be implicit), we define the predicate as $f_{P,Q}(R_1, R_2) = 1$ iff $\log_P R_1 = \log_Q R_2$. Now, verifying the equality is just an instance of the decisional Diffie-Hellman problem, while computing R_1 from P, Q, and R_2 requires solving the computational Diffie-Hellman problem. On the other hand, knowing the trapdoor (i.e., $\log_P Q$) makes this last problem easy. In more detail:

- $I(1^k)$ runs $\mathcal{IG}(1^k)$ to obtain $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$. Then, it chooses random $P \in \mathbb{G}_1$ and random $\alpha \in \mathbb{Z}_q^*$ (recall that q is the order of $\mathbb{G}_1, \mathbb{G}_2$). It sets $Q = \alpha P$ and outputs the index $i = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q)$ and the trapdoor α.
- $D(i)$ chooses random $\beta \in \mathbb{Z}_q$ and outputs $(\beta P, \beta Q)$.
- $F(i, R_1, R_2)$ (with $R_1, R_2 \in \mathbb{G}_1$) outputs 1 iff $\hat{e}(P, R_2) = \hat{e}(Q, R_1)$.
- $F^{-1}(\alpha, R_2)$ outputs $\alpha^{-1} R_2$.

It is immediate from the discussion earlier that the above forms a publicly-verifiable trapdoor predicate if $\mathcal{IG}$ satisfies the computational BDH assumption (since the computational BDH assumption for $\mathcal{IG}$ implies that the computational Diffie-Hellman problem in $\mathbb{G}_1$ is hard).

It is furthermore not difficult to see (following, e.g., Section 4.10.2 of [22]) that publicly-verifiable trapdoor predicates satisfying some additional assumptions are sufficient to implement the "hidden-bits paradigm" [22, Definition 4.10.3] (and hence NIZK) in the common random string model. These additional assumptions, informally, relate to:

1. The ability to efficiently recognize elements of the index set $\bar{I}$, or to prove that a given i is indeed in $\bar{I}$ (see [7]).
2. The existence of a sampling algorithm D' which, on input $i \in \bar{I}$ and random coins ω, outputs a uniformly-distributed element $y \in D_i$ and furthermore has the following property: for all PPT algorithms A the following is negligible in k:

$$\Pr[(i, td) \leftarrow I(1^k); \omega \leftarrow \{0,1\}^*; y \leftarrow D'(i; \omega); x \leftarrow A(1^k, i, y, \omega) : f_i(x, y) = 1];$$

   i.e., it is hard to find a valid "match" even given the random coins of D'. (Trapdoor permutations satisfying a notion analogous to the above are called "enhanced." The reader is referred to Appendix C.1 of [23], which corrects Section 4.10.2 of [22], for discussion.)

Although these assumptions seem plausible for BDH parameter generators used in practice, we do *not* require these assumptions for our desired application to CCA2 security as discussed in Section 4.3. In particular, since for our desired application the receiver — and not a third party — establishes the "public parameters," NIZK in the common *reference* string model (as opposed to the common random string model) is sufficient. This enables a number of simplifications. In particular, the receiver can simply generate parameters $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P)$ and publish these values as part of its public key along with a sufficiently-long sequence $R_1, \ldots, R_n$ of randomly-generated values in $\mathbb{G}_1$ which will serve as the common reference string. When proving a statement, a sender chooses random $\alpha \in \mathbb{Z}_q^*$, computes $Q = \alpha P$, and sends Q, thereby defining an index $i = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q)$ for the publicly-verifiable trapdoor predicate introduced earlier. Note that since the sender has the trapdoor α, he may indeed implement the "hidden-bits paradigm," as desired.

B Proof of Theorem 3

The proof is a relatively straightforward adaptation of [20]. Given a PPT adversary A, we introduce a sequence of games where the first game Game_0 corresponds to the experiment of Definition 3 while in the final game the view of A is independent of the bit b. For each pair of consecutive games in the sequence, we argue that the difference between the probability that $b' = b$ in the first game and the probability that $b' = b$ in the second game is negligible. Since there are only a constant number of games, and the probability that $b' = b$ in the final game is exactly $1/2$, this completes the proof.

Let p_0 denote the probability that $b' = b$ in Game_0, as described in Definition 3 (technically, p_0 is a function of the security parameter k but we do not explicitly write this). Let w^* be the "target" node chosen by A, let (M_0, M_1) denote the "challenge messages" submitted by A, and let $C^* = \langle C_1^*, C_2^* \rangle$ denote the "challenge ciphertext" received by A where:

$$C_1^* = \mathsf{Enc}(PK, w^*, \sigma^*; H(w^*, \sigma^*, M_b)) \qquad \text{and} \qquad C_2^* = G(\sigma^*) \oplus M_b,$$

for randomly-chosen σ^* and b. Game_1 is exactly the same as Game_0 except that whenever A (after receiving the challenge ciphertext) requests decryption of a ciphertext $\langle C_1^*, C_2 \rangle$ by a node w, this query is answered by $\perp$. Let p_1 denote the probability that $b' = b$ in Game_1.

We claim that $|p_0 - p_1|$ is negligible. To prove this we argue that, with all but negligible probability, all decryption requests by A of the form considered above are answered by $\perp$ in Game_0 anyway. To see this, consider any request by A for node w to decrypt ciphertext $\langle C_1^*, C_2 \rangle$. Note that we must have $(w, C_2) \neq (w^*, C_2^*)$. Let $\sigma \stackrel{\text{def}}{=} \mathsf{Dec}(PK, w, SK_w, C_1^*)$, and let $M \stackrel{\text{def}}{=} G(\sigma) \oplus C_2$. If $w = w^*$ then $\sigma = \sigma^*$; but then $C_2 \neq C_2^*$ implies that $M \neq M_b$. In any case, then, we have $(w, \sigma, M) \neq (w^*, \sigma^*, M_b)$. Since $\mathcal{BTE}$ satisfies the "unique randomness" property (see Section 3.2), the only way this decryption query will not be answered with $\perp$ is in case $H(w, \sigma, M) = H(w^*, \sigma^*, M_b)$. Since the output length of H is super-logarithmic (this is implied by the SN-CPA security of $\mathcal{BTE}$), the probability that this occurs is negligible.[7] Applying a union bound over all oracle queries of A (specifically, A's decryption queries, queries to H, and challenge query) proves the stated claim.

In Game_2 we again modify the way decryption requests of A are handled. In particular, for all decryption queries of A not covered by the rule stated above (namely, whenever A requests that a node w decrypt ciphertext $\langle C_1, C_2 \rangle$, and either this is before A has received the challenge ciphertext or else $C_1 \neq C_1^*$) we proceed as follows: For each query $H(w_i, \sigma_i, M_i)$ made by A to its random oracle H, with corresponding answer r_i, we check whether: (1) $w_i = w$; and (2) $\mathsf{Enc}(PK, w, \sigma_i; r_i) = C_1$. If there exists such a tuple (w_i, σ_i, M_i) satisfying the above (we call this a "match"), then this decryption query of A is answered by computing $M' = G(\sigma_i) \oplus C_2$ and outputting M' iff $\mathsf{Enc}(PK, w, \sigma_i; H(w, \sigma_i, M')) = C_1$. Otherwise, the decryption query is answered with $\perp$. Let p_2 denote the probability that $b' = b$ in Game_2.

We claim that $|p_2 - p_1|$ is negligible. Clearly, whenever a "match" is found in Game_2, the corresponding decryption query of A is answered identically to how this query would be answered in Game_1. Thus we only need to argue that, with all but negligible probability, when a "match" is *not* found in Game_2 the decryption query would have been answered with $\perp$ in Game_1, anyway. To see this, consider a request by A for node w to decrypt $\langle C_1, C_2 \rangle$. Let $\sigma = \mathsf{Dec}(PK, w, SK_w, C_1)$ (if $\sigma = \perp$ we are done, so assume otherwise), and let $M = G(\sigma) \oplus C_2$. By the unique randomness property of $\mathcal{BTE}$, there is at most one r for which $C_1 = \mathsf{Enc}(PK, w, \sigma; r)$. Since no match was found, it is either the case that A asked the query $H(w, \sigma, M)$ but the response to this query was

[7] Without the unique randomness assumption, we would need to argue that the set of coins r for which $C_1^* = \mathsf{Enc}(PK, w, \sigma; r)$ constitutes a negligible fraction of all possible random coins. While this seems easy to prove if we assume security of the encryption scheme against *non-uniform* adversaries, it appears difficult to prove otherwise.

not r, or A did not ask this query. In the former case, the decryption query will be rejected, while in the latter case it will be rejected with all but negligible probability. Applying a union bound as before proves the stated claim.

In Game_3, we let the second component of the challenge ciphertext (i.e., C_2^*) be a randomly chosen string of the appropriate length. Let p_3 denote the probability that $b' = b$ in Game_3; clearly $p_3 = 1/2$. To complete the proof of the theorem, we thus only need to argue that $|p_3 - p_2|$ is negligible. Note that the only difference between the two games — from the point of view of A — occurs in case A makes a query $G(\sigma^*)$ (where σ^* represents the random value used to construct the challenge ciphertext). Let q denote the probability that A makes such a query. We claim that q is negligible since $\mathcal{BTE}$ is secure in the sense of SN-CPA. Indeed, consider the following adversary $B(1^k, 1^\ell)$ attacking $\mathcal{BTE}$ in the sense of SN-CPA:

1. Run $A(1^k, 1^\ell)$ to obtain a target node w^*. This same target node is output by B.
2. B obtains a public key PK and secret keys $\{SK_w\}$ as in Definition 2. B gives these to A.
3. B simulates random oracles H, G for A, and decryption queries of A are answered as in Game_2.
4. When A queries $\mathsf{challenge}(M_0, M_1)$, B chooses σ_0 uniformly at random and sets σ_1 to be an arbitrary constant (say, the all-0 string). B queries $\mathsf{challenge}(\sigma_0, \sigma_1)$ and receives a ciphertext C_1^*. Next, B chooses C_2^* uniformly at random and gives $\langle C_1^*, C_2^* \rangle$ to A.
5. Decryption queries of A are again answered as in Game_2.
6. Furthermore, if at any point A makes a query $G(\sigma_0)$ then B outputs "0" and stops. Otherwise, if the experiment ends without A having made such a query, B outputs "1".

Note that if C_1^* is an encryption of σ_0 then, from the point of view of A, the above experiment is identical to games $\mathsf{Game}_2/\mathsf{Game}_3$ until the point in time (if any) that A queries $G(\sigma_0)$ (and this occurs with probability q). On the other hand, if C_1^* is an encryption of σ_1 then A has no information about σ_0 and hence the probability that A queries $G(\sigma_0)$ is some negligible quantity negl (recall that $|\sigma_0| = k$). Thus, the advantage of B (in attacking $\mathcal{BTE}$ in the sense of SN-CPA) is

$$\left| \frac{q}{2} + \frac{1 - \mathsf{negl}}{2} - \frac{1}{2} \right| = \left| \frac{q - \mathsf{negl}}{2} \right|,$$

and so q must be negligible, as desired. This completes the proof of Theorem 3.