A Forward-Secure Public-Key Encryption Scheme

Ran Canetti[*] Shai Halevi[*] Jonathan Katz[†]

May 1, 2003

Abstract

Cryptographic computations are often carried out on insecure devices for which the threat of key exposure represents a serious and realistic concern. In an effort to mitigate the damage caused by exposure of secret keys stored on such devices, the paradigm of *forward security* was introduced. 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. A number of constructions of forward-secure digital signature schemes, key-exchange protocols, and symmetric-key schemes are known.

We present the first non-trivial constructions of (non-interactive) forward-secure public-key encryption schemes. Our main construction achieves security against chosen-plaintext attacks under the decisional bilinear Diffie-Hellman assumption in the standard model. This scheme is practical, and all parameters grow at most logarithmically with the total number of time periods. We also give a slightly more efficient scheme in the random oracle model. Both our schemes can be extended to achieve security against chosen-ciphertext attacks and to support an unbounded number of time periods.

Toward our goal, we introduce the notion of *binary tree encryption* (related to hierarchical identity-based encryption) and show how to construct a binary tree encryption scheme in the standard model. This new primitive may be of independent interest.

Key words: Bilinear Diffie-Hellman, Encryption, Forward security, Key exposure.

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

[†]Dept. of Computer Science, University of Maryland, College Park, MD. Portions of this work were done while at Columbia University. jkatz@cs.umd.edu.

Contents

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, even PCs), new techniques are needed to deal with this concern.

A variety of methods have been introduced in an attempt to deal with this threat (including secret sharing [37], threshold cryptography [14], and proactive cryptography [34]). 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 [22] and Diffie, et al. [15]: a forward-secure key-exchange protocol guarantees that exposure of long-term secret information does not compromise the security of previously-generated session keys. We remark that a forward-secure key-exchange protocol naturally gives rise to a forward-secure *interactive* encryption scheme in which the sender and receiver generate a shared key which is used to encrypt a single message and is then promptly erased.

Subsequently, Anderson [3] suggested forward security for the more challenging *non-interactive* setting. The lifetime of the system is divided into N intervals (or time periods) labeled $0, \ldots, N-1$. The receiver initially stores secret key SK_0 and this secret key "evolves" with time. Namely, at the beginning of time period i, the receiver 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. The public (encryption) key remains fixed throughout the lifetime of the scheme; this is crucial for making such a scheme viable. A forward-secure encryption scheme guarantees that even if an adversary learns SK_i (for some i), messages encrypted during all time periods *prior* to i remain secret (a formal definition is given in Section 3). Note that since the adversary obtains all secrets existing at time i, the model inherently cannot protect the secrecy of messages encrypted at time i and at all subsequent time periods.

A number of constructions of forward-secure signature/identification schemes are known [6, 29, 1, 25, 31, 28], and forward security for non-interactive, symmetric-key encryption has also been studied [7]. 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 the obvious practical advantage that a compromise of the system does not compromise the secrecy of previously-encrypted information; it is thus appropriate for devices operating in insecure environments. Furthermore, using such a scheme enables provable security against *adaptive* adversaries who may choose which parties to corrupt based on information learned in the course of a given protocol; we comment further on this below.

1.1 Our Contributions

Forward secure encryption. We define a rigorous notion of security for forward-secure public-key encryption and give efficient constructions of schemes satisfying this notion. We prove semantic security of our main scheme in the standard model based on the decisional version of the bilinear Diffie-Hellman (BDH) assumption (cf. [26, 10]). All salient parameters of this scheme are 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

	Standard model	Random oracle model
Key generation time	$\mathcal{O}(\log N)$	$\mathcal{O}(1)$
Encryption/Decryption time	$\widetilde{\mathcal{O}}(\log N)$	$\mathcal{O}(\log N)$
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: Summary of dependencies on the total number of time periods N.

the random oracle model[1] under the *computational* BDH assumption (which is presumably weaker than the decisional 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 [10] and may therefore be practical for reasonable values of N.

We also note 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 (i.e., the number of time periods need not be known when the public key is generated and published). This has the added advantage that the efficiency and security of these schemes depend only on the number of time periods elapsed thus far. We also sketch two ways to extend our schemes to achieve security against adaptive chosen-ciphertext attacks [35, 8]. In the standard model, we note that the techniques of Sahai [36] using NIZK proofs (and based on earlier work of Naor and Yung [32]) extend to our setting; interestingly, we show also that general NIZK proofs may be implemented based on the BDH assumption alone (so that we do not require the additional assumption of trapdoor permutations). In the random oracle model, we modify the Fujisaki-Okamoto transformation [18] for our purposes and thereby obtain a secure scheme.

Other contributions. Our constructions are based on the hierarchical identity-based encryption (HIBE) scheme of Gentry and Silverberg [19] which, in turn, is based on the identity-based encryption scheme of Boneh and Franklin [10]. As a first step toward our construction, 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. We believe the BTE primitive is interesting in its own right, and it may lead to many other applications; as an example, we show in Appendix A how a full-blown HIBE scheme may be based on any BTE scheme.

At a high level, our construction shares similarities with previous tree-based, forward-secure signature schemes (e.g., those of [6, 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. We note that this tree traversal technique can also be used to improve the efficiency of these algorithms in the tree-based signature schemes mentioned above, from $O(\log N)$ to $O(1)$.

Applications. In addition to the obvious application of forward-secure PKE to protecting against

[1]We stress that a proof in the random oracle model provides no guarantee as to the security of the protocol once the random oracle is instantiated with an efficiently-computable "cryptographic hash function". A proof in the random oracle model should only be regarded as a heuristic argument indicating that the construction is secure.

key exposure, we note that it may also be used to drastically improve the efficiency of non-interactive *adaptively-secure* encryption in the context of secure multi-party protocols. In such protocols, an adaptive adversary may choose to corrupt a player at some point in the protocol *after* that player has already received several encrypted messages. Learning the player's secret key will (in general) allow the adversary to read all past messages, thereby making it much harder to prove any simulation-based notion of security. In all known adaptively-secure non-interactive encryption schemes (e.g., [4, 11, 5, 13]) the size of the decryption key must exceed the total length of all messages to be decrypted throughout the lifetime of the system. Furthermore, Nielsen has recently shown that this property is *essential* for encryption schemes that are not key-evolving [33] (this holds even if the model itself allows data erasures). Forward-secure encryption enables us to circumvent this lower bound by having each player update its (short) key after every message is sent. Now, an adversary who corrupts a player will be unable to read messages sent to this player in previous rounds and a simulation-based proof of security is thereby possible.

1.2 Organization

In Section 2 we define BTE and provide a construction which is provably secure under the decisional BDH assumption (described in Section 2.1) in the standard model. In this 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 (as mentioned above). In Section 3, we formally define forward-security for public-key encryption and show how a forward-secure PKE scheme can be constructed from any BTE scheme. Combining these results, we obtain a forward-secure PKE scheme with the advertised parameters.

2 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 relaxed version of hierarchical identity-based encryption (HIBE) [24, 19]. In addition to being an essential step in our construction of forward-secure encryption, we believe BTE is independently interesting: in particular, we show in Appendix A how to implement a full-blown HIBE from BTE; since we describe a BTE whose security can be proven in the standard model, this implies a secure HIBE in the standard model (albeit, with a weaker notion of security than that considered in [19]).

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 destined 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 $w1$. (Recall that in HIBE the tree can have arbitrary degree, and a child of node v can be labeled $v.s$ for any arbitrary string s.) A functional definition follows.

Definition 1 A (public-key) binary tree encryption (BTE) scheme is a 4-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 ℓ for the depth of the tree. It returns a master public key PK and an initial (root) secret key SK_ε. (We assume that the values of k and ℓ are implicit in PK and all node secret keys.)

- 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.

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

The security notion that we present here for BTE requires the attacker to commit to the node to be attacked *in advance* (i.e., before seeing the public key); we call this attack scenario a *selective-node (SN) attack* (cf. "selective forgery" of signatures [23]). While our definition is weaker than the corresponding definition for HIBE achieved by [19], our definition suffices to construct a forward-secure PKE scheme from any BTE scheme (cf. Section 3).

Definition 2 A BTE scheme is *secure against selective-node, chosen-plaintext attacks* (SN-CPA) if for all polynomially-bounded functions $\ell(\cdot)$, the advantage of any PPT adversary A in the following game is negligible in the security parameter:[2]

1. $A(1^k, \ell(k))$ outputs a name $w^* \in \{0,1\}^{\leq\ell(k)}$ of a node.
2. Algorithm $\mathsf{Gen}(1^k, \ell(k))$ 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), and also the secret keys for the two children of w^* (if $|w^*| < \ell$). 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);

   – $w = w^*0$ or $w = w^*1$ (i.e., w is a child of w^*; this is only when $|w^*| < \ell$).

   (Note that this allows the adversary to compute $SK_{w'}$ for any node $w' \in \{0,1\}^{\leq\ell(k)}$ 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. ■

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* (SN-CCA) if for all polynomially-bounded functions $\ell(\cdot)$, the advantage of any PPT adversary A in the following game is negligible in the security parameter:

1. $A(1^k, \ell(k))$ outputs a name $w^* \in \{0,1\}^{\leq\ell(k)}$ of a node.
2. Algorithm $\mathsf{Gen}(1^k, \ell(k))$ outputs (PK, SK_ε). The adversary is given PK and node secret keys as in Definition 2.

[2]Recall that a function is negligible if it approaches zero faster than any inverse polynomial.

3. The adversary may query a decryption oracle $\mathsf{Dec}^*(\cdot,\cdot)$. On query $\mathsf{Dec}^*(w, C)$ with $w \in \{0,1\}^{\leq \ell(k)}$, a key SK_w is derived from SK_ε (if one was not derived previously) 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 $w \neq w^*$).

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. ■

2.1 The Bilinear Diffie-Hellman Assumption

The security of our BTE scheme is based on the difficulty of the bilinear Diffie-Hellman (BDH) problem as formalized by Boneh and Franklin [10] (see also [26]). The computational version of this assumption has been used for a number of cryptographic constructions; furthermore, the decisional version of the assumption (called the BDDH assumption in [10]) is also believed to hold. We review the relevant definitions as they appear in [10, 19]. 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 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 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 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:

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

The *decisional BDH problem* is to distinguish between tuples of the form $(P, \alpha P, \beta P, \gamma P, \alpha\beta\gamma P)$ and $(P, \alpha P, \beta P, \gamma P, \mu P)$ for random $P, \alpha, \beta, \gamma, \mu$. Formally, we say $\mathcal{IG}$ *satisfies the decisional BDH assumption* if the following probability is negligible (in k) for all PPT algorithms A:

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

The decisional BDH assumption immediately implies that it is computationally infeasible to distinguish between tuples of the form $(P, \alpha P, \beta P, \gamma P, \hat{e}(P,P)^{\alpha\beta\gamma})$ and $(P, \alpha P, \beta P, \gamma P, r)$ for random $P \in \mathbb{G}_1$, $\alpha, \beta, \gamma \in \mathbb{Z}_q$, and $r \in \mathbb{G}_2$.

BDH parameter generators believed to satisfy the above assumptions can be constructed from Weil and Tate pairings associated with super-singular elliptic curves or Abelian varieties. As our results do not depend on any specific instantiation, we refer the interested reader to [10] for details.

2.2 A BTE Scheme 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.*

We describe such a scheme now. The starting point for our construction is the HIBE scheme of Gentry and Silverberg [19]. 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 [19] may be used to implement a secure BTE in the random oracle model for trees of constant depth.) The HIBE of Gentry and Silverberg (as well as the IBE scheme of Boneh and Franklin [10]) uses random oracles in three ways: to derive random elements from identities, 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 easily be avoided if one is willing forgo chosen-ciphertext security and to use the *decisional* BDH assumption. More interestingly, we show below that the random oracle mapping identities to group elements can be replaced by an $O(\ell)$-wise independent hash function (recall from Definition 2 that ℓ is the depth of the tree). Additionally, a proof of security may be obtained even for trees of polynomial depth.

Notation and conventions. In the description below we let ℓ denote the depth of the tree. The i-bit prefix of a string $w_1 w_2 \ldots w_t$ is denoted by $w|_i$. Namely, $w|_i = w_1 \ldots w_i$. In our scheme, we use a $(2\ell+1)$-wise independent family $\mathcal{H}$ of functions $H : \{0,1\}^{\leq \ell} \to \mathbb{G}_1$ for which, given elements $x_1, \ldots, x_k \in \{0,1\}^{\leq \ell}$ and $g_1, \ldots, g_k \in \mathbb{G}_1$ (with $k \leq 2\ell + 1$), it is possible to efficiently sample a random $H \in \mathcal{H}$ satisfying $H(x_i) = g_i$ for $i = 1, \ldots, k$. One possible instantiation is to let $\mathcal{H} = \{H_{h_0,\ldots,h_{2\ell}}(x)\}_{h_0,\ldots,h_{2\ell} \in \mathbb{G}_1}$, where $H_{h_0,\ldots,h_{2\ell}}(x) \overset{\text{def}}{=} h_0 + \tilde{x}h_1 + \cdots + \tilde{x}^{2\ell}h_{2\ell}$ and $\tilde{x}$ represents some fixed one-to-one encoding of $x \in \{0,1\}^{\leq \ell}$ as an element in $\mathbb{Z}_q$ (this requires that $2^{\ell+1} \leq q$; alternately, we can include in the public key a universal one-way hash function mapping $\{0,1\}^{\leq \ell}$ to $\mathbb{Z}_q$). We let ε denote the empty string. Finally, $\mathcal{IG}$ is a BDH parameter generator believed to satisfy the decisional BDH assumption.

$\mathsf{Gen}(1^k, \ell)$ 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 $H \in \mathcal{H}$ (note: ℓ is implicit in $\mathcal{H}$).
4. The public key is $PK = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, \ell, H)$. The root secret key is $SK_\varepsilon = \alpha H(\varepsilon)$.

In general, for $w = w_1 \ldots w_t$, the secret key of node w will consist of $t+1$ group elements; these are denoted by $SK_w = (R_{w|_1}, R_{w|_2}, \ldots, R_{w|_{t-1}}, R_w, S_w)$ (for the special case of $w = \varepsilon$ we simply have $SK_\varepsilon = S_\varepsilon = \alpha H(\varepsilon)$). With this in mind, we now describe the key derivation algorithm.

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

1. Let $w = w_1 \ldots w_t$. Parse SK_w as $(R_{w|_1}, R_{w|_2}, \ldots, R_{w|_{t-1}}, R_w, S_w)$.
2. Choose random $\rho_{w0}, \rho_{w1} \in \mathbb{Z}_q$. Set $R_{w0} = \rho_{w0}P$, $R_{w1} = \rho_{w1}P$, $S_{w0} = S_w + \rho_{w0}H(w0)$, and $S_{w1} = S_w + \rho_{w1}H(w1)$.
3. Output $SK_{w0} = (R_{w|_1}, \ldots, R_w, R_{w0}, S_{w0})$ and $SK_{w1} = (R_{w|_1}, \ldots, R_w, R_{w1}, S_{w1})$.

Slightly better efficiency is possible — without affecting the security of the scheme — by setting $\rho_{w0} = \rho_{w1}$. Doing so will also result in a slightly shorter secret key when this scheme is used to construct a forward-secure PKE scheme in the following section.

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

1. Let $w = w_1 \ldots w_t$. 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 $(R_{w|_1}, \ldots, R_w, S_w)$, and parse C as $(U_0, U_1, \ldots, U_t, V)$.
2. Output $M = V/d$, where
$$d = \frac{\hat{e}(U_0, S_w)}{\prod_{i=1}^t \ \hat{e}(R_{w|_i}, U_i)}.$$

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,

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

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 against SN-CPA.*

Proof Let $\ell(\cdot)$ be some function which is polynomially-bounded; in what follows, we simply write ℓ instead of $\ell(k)$. Assume a PPT adversary A attacking the above scheme in the SN-CPA attack scenario, and denote the probability that A succeeds by $\Pr_A[\mathsf{Succ}]$. We construct a new adversary B which 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 only in their last bit.)

The new adversary B is given the output $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ of $\mathcal{IG}(1^k)$ and also $(P, Q = \alpha P, I_\varepsilon = \beta P, U_0 = \gamma P, V' = \mu P)$; the goal of B (informally) is to determine whether $\mu = \alpha\beta\gamma$ or not. For that purpose, B attempts to simulate an instance of the encryption scheme for A as follows: B initiates a run of A, and A commits to the target node $w^* = w_1^* w_2^* \ldots w_t^*$ (with $t \le \ell$). Now, for $1 \le i \le t$, B chooses χ_i, λ_i, and φ_i at random from $\mathbb{Z}_q$. If $t < \ell$, then for $b \in \{0,1\}$ it also chooses λ_{t+1}^b and φ_{t+1}^b 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 $\mathcal{H}$ 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 $H(w^*0) = \lambda^0_{t+1} P - \frac{1}{\varphi^0_{t+1}} I_\varepsilon$ and $H(w^*1) = \lambda^1_{t+1} P - \frac{1}{\varphi^1_{t+1}} I_\varepsilon$.

Note that since there are at most $2\ell+1$ constraints, B can choose such a (random) $H \in \mathcal{H}$ efficiently. Furthermore, because $\mathcal{H}$ is a $(2\ell+1)$-independent family of functions, this choice of H is distributed identically to H in the real experiment. B sets $PK = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, \ell, H)$ 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 secret keys for any node w in the tree such that w is not a prefix of w^*. To generate these secret keys, B chooses (for $1 \le i \le t$) $R_{w^*|_i} \in \mathbb{G}_1$ at random and furthermore sets $R_{w^*|_{\bar{i}}} = \varphi_i Q$. Next, for all $1 \le i \le t$, B sets

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

(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 additionally sets (for $b \in \{0,1\}$) $R_{w^*b} = \varphi^b_{t+1} Q$ and $S_{w^*b} = \lambda^b_{t+1} \varphi^b_{t+1} Q + \sum_{j=1}^{t} \chi_i R_{w^*|_{\bar{j}}}$. 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. For $1 \le i \le t$ let $\rho_{w^*|_i} \in \mathbb{Z}_q$ be such that $R_{w^*|_i} = \rho_{w^*|_i} P$ with an analogous definition for $\rho_{w^*|_{\bar{i}}}$; note that $\rho_{w^*|_{\bar{i}}} = \varphi_i \alpha$. Clearly, these ρ-values are all independently and uniformly distributed in $\mathbb{Z}_q$. Furthermore, in a real execution of the SN-CPA experiment we would have $S_w = \alpha H(\varepsilon) + \sum_{j=1}^{|w|} \rho_{w|_j} H(w|_j)$ for any w. For $w = w^*|_{\bar{i}}$ this means

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

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

Continuing its execution of A (after having provided the appropriate secret keys), B responds to the query challenge(M_0, M_1) by choosing a random bit b and returning

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

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 re-write 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 at most 1/2. The advantage of B is therefore at least $|\Pr_A[\mathsf{Succ}]-1/2|$; since the advantage of B is negligible (by assumption on $\mathcal{IG}$), the advantage of A must be negligible as well. This concludes the proofs of Proposition 1 and Theorem 1. ■

Scheme parameters. In the construction above, a secret key of node w at level t consists of $t+1$ elements of $\mathbb{G}_1$. However, we may notice that all elements of the secret key except for R_w and S_w already appear in the secret key of the parent of w. Thus, the secret keys of all the nodes on the path from the root to w can be stored using only $O(t)$ group elements (we make use of this fact when constructing our forward-secure PKE in the following section).

The key-generation algorithm requires time linear in ℓ (the depth of the tree), where this complexity is due to selection of the hash function H. The key-derivation algorithm requires a constant number of operations in $\mathbb{G}_1$ and two applications of the hash function H; a single evaluation of H requires $O(\ell)$ time when H is an $O(\ell)$-degree polynomial as suggested above. Encryption for a node at level t requires t applications of H, $t+1$ multiplications in $\mathbb{G}_1$, one application of $\hat{e}$, and one multiplication and one exponentiation in $\mathbb{G}_2$. (Note that one can evaluate H on t distinct points (for $t \leq \ell$) in time $O(\ell \log^2 \ell)$ — rather than the naïve $O(t\ell)$ — when H is a $2\ell+1$-degree polynomial [2, Section 8.5].) Decryption by a node at level t requires $t+1$ evaluations of $\hat{e}$ along with t operations in $\mathbb{G}_2$.

Applications to HIBE. One can construct a full-blown HIBE scheme from any BTE scheme simply by encoding in binary all the identities in the system using collision-resistant hashing at every level of the hierarchy. (The security obtained, however, is weaker than that guaranteed by [19].) See Appendix A for details.

Construction in the random oracle model. The scheme above remains secure if the hash function 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 function ℓ.) Instantiating H in this way improves several of the scheme parameters: the public-key size, key-generation time, and key-update time are now independent of ℓ, and encryption now takes time $O(\ell)$ (as opposed to $\tilde{O}(\ell)$).

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 rather than the decisional version[3]: simply replace the component $M \cdot \hat{e}(Q, H(\varepsilon))^r$ of the ciphertext by $M \oplus H'(\hat{e}(Q, H(\varepsilon))^r)$, where $H' : \mathbb{G}_2 \to \{0,1\}^n$ is modeled as an independent random oracle.

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 extend to our setting the techniques of Sahai [36] based on earlier work of Naor and Yung [32] (see also [30]); namely, we use *simulation-sound* NIZK proofs to achieve chosen-ciphertext security. In more detail (we assume the reader is familiar with [36, 30]), 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 computes $C_1 \leftarrow \mathsf{Enc}(PK_1, w, M)$, $C_2 \leftarrow \mathsf{Enc}(PK_2, w, M)$, and a simulation-sound NIZK proof of consistency π using string r (explained in more detail below); 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 language L defined by:

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

[3]Of course, 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.

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 $w' \overset{?}{=} w$ and whether π is an accepting 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 $\perp$. We omit the tedious details (which follow [36, 30]) that this scheme is indeed secure in the sense of SN-CCA.

It is known that simulation-sound NIZK (admitting efficient provers) may be based on the assumption of trapdoor permutations [36, 17]. We observe, additionally, that simulation-sound NIZK in the common reference string model[4] may also be based on the (seemingly incomparable) decisional BDH assumption. To see this, note that Sahai's construction [36] of simulation-sound NIZK may be based on any (standard) NIZK proof system. These, in turn, may be constructed using the "hidden-bits paradigm" set forth in [17] (see also [20, Section 4.10.2]). Our key observation (see Appendix B) is that this "hidden-bits paradigm" (which is usually achieved using trapdoor permutations) may be implemented using *publicly-verifiable trapdoor predicates* — a generalization of trapdoor permutations considered previously [16] and formally introduced here. We furthermore show in Appendix B how the decisional BDH assumption naturally gives rise to a publicly-verifiable trapdoor predicate. Combining these results yields the following theorem:

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 by applying, e.g., a secure variant of the Fujisaki-Okamoto transformation [18] (note that the Fujisaki-Okamoto transformation only applies to standard PKE and must be appropriately modified for the case of BTE). We propose the following scheme: Let $(\mathsf{Enc}', \mathsf{Dec}')$ represent any *symmetric-key* encryption scheme secure in the sense of IND-P0-C2 (cf. [27]), let Enc denote (the encryption algorithm for) a BTE scheme secure in the sense of SN-CPA which encrypts messages at least as long as the security parameter, and let H and G denote independent random oracles which are also independent of any random oracles used by Enc or Enc'. Consider then the following BTE scheme Enc'':

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

where σ is a randomly-chosen value and the notation $\mathsf{Enc}(\cdot, \cdot, \cdot; r)$ denotes that random coins r are used when performing the stated encryption. Key generation and key derivation are done in the obvious way. 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 $M = \mathsf{Dec}'_{G(\sigma)}(C_2)$. If $C_1 \overset{?}{=} \mathsf{Enc}(PK, w, \sigma; H(w, \sigma, M))$, the output is M; otherwise, the output is $\perp$. Security of this modified scheme is given by the following theorem, whose proof exactly follows that of [18]:

Theorem 3 *If* Enc *is a BTE scheme secure in the sense of SN-CPA, and* Enc' *is a symmetric-key encryption scheme secure in the sense of IND-P0-C2, then the above construction yields a BTE scheme secure in the sense of SN-CCA in the random oracle model.*

[4]In the common reference string model — to be distinguished from the common random string model — the reference string r may be chosen from an arbitrary, poly-time computable distribution (and not necessarily a uniform one). Since the string r included with the public key is generated by the recipient (and not by some third party), working in the common reference string model is sufficient for our purposes.

3 Forward-Secure Public-Key Encryption

In this section, 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 — which is an immediate application of the BTE primitive discussed in the previous section — we then describe a construction of a forward-secure scheme all of whose parameters grow at most *logarithmically* with the total number of time periods.

3.1 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 [6]; the latter, however, is new and requires some care.

Definition 4 A (public-key) *key-evolving encryption* ($\mathsf{ke\text{-}PKE}$) *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.
- The *key update algorithm* Upd takes as input PK, an index $i < N$ 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 $i \leq N$ of a time period, and a message M. It returns a ciphertext C.
- The *decryption algorithm* Dec takes as input PK, an index $i \leq N$ of the current time period, the associated secret key SK_i, and a ciphertext C. It returns a message M.

We make the standard 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 $M = \mathsf{Dec}(PK, i, SK_i, \mathsf{Enc}(PK, w, M))$. ■

Our definitions of forward-secure public-key encryption generalize the standard notions of security for PKE, similar to the way in which the definitions of [6] generalize the standard notion of security for signature schemes.

Definition 5 A ke-PKE scheme is *forward-secure against chosen plaintext attacks* ($\mathsf{fs\text{-}CPA}$) if for all polynomially-bounded functions $N(\cdot)$, the advantage of any PPT adversary in the following game is negligible in the security parameter:

Setup: $\mathsf{Gen}(1^k, N(k))$ 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 $\mathsf{Upd}(PK, i-1, \cdots \mathsf{Upd}(PK, 0, SK_0) \cdots)$. 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* (fs-CCA) if for all polynomially-bounded functions $N(\cdot)$, the advantage of any PPT adversary in the following game is negligible in the security parameter:

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 $0 \leq j < i < N$ and $0 \leq k < N$. These queries are answered as follows:

- The breakin and challenge queries are answered as in Definition 3.1.
- On query $\mathsf{Dec}^*(k, C)$, the appropriate key SK_k is first derived using SK_0 and PK. Then, the adversary is 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 $k \neq j$, *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. ■

Remark 1: 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 queries 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 an N-fold loss 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.

Remark 2: Relaxing chosen-ciphertext security. Note that Definition 6 allows the adversary to make decryption queries for various time periods in arbitrary order (not necessarily chronological). Furthermore, the adversary is allowed to obtain the decryption of the challenge ciphertext C^*, as long as the decryption is for a different time period than the one in which the ciphertext was generated. This extra power given to the adversary results in a definition that is probably stronger than what is needed in most settings, and can potentially be relaxed and still provide adequate security. Still, we present this notion since it is the strongest natural interpretation of the CCA paradigm and the CCA-secure variants of our scheme achieve it.

3.2 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 $\{(sk_i, pk_i)\}$ 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 an improved solution can be built from an *identity-based* encryption scheme. Here, the public key is the "master public key" of the identity-based scheme, and SK_i is computed as the "personal secret key" of a user with identity i (the scheme is otherwise identical to the above). This solution achieves $O(1)$ public key size, but still has $O(N)$ secret-key size and key-generation time.

In fact, one can improve this last solution somewhat: instead of a large secret key, it is enough if the user keeps a large *non-secret* file containing one record per period. The record for period i contains the secret key SK_i encrypted under the public key for time period $i-1$. At the beginning of period i, the user obtains record i, uses key SK_{i-1} to recover SK_i, and then erases SK_{i-1}. This solution achieves essentially the same efficiency as the "simple forward-secure signatures" of Krawczyk [29] (and in particular requires $O(N)$ non-secret storage and key-generation time).

3.3 A Construction with Logarithmic Complexity

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 the previous section.

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 $N-1$), simply use a BTE of depth ℓ and associate the time periods with all nodes of the tree according to a pre-order traversal. (Let w^i denote the node associated with period i. In a pre-order traversal, $w^0 = \varepsilon$ and if w^i is an *internal node* then $w^{i+1} = w^i0$. 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 is simply the root public key for the BTE scheme; the secret key for period i consists of the secret key for node w^i as well as those 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 period 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; 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 those for all right siblings of the nodes on the path from the root to w^{i+1}.

Our method of associating time periods with nodes of a binary tree is reminiscent of previous tree-based forward-secure signature schemes [6, 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 efficiency improvement from $O(\log N)$ to $O(1)$ in the key-generation and key-update times. We remark that our tree-traversal method can also be applied to the signature schemes of [6, 1, 31] with similar efficiency gain.

More formally, given a BTE scheme $(\mathsf{Gen}, \mathsf{Der}, \mathsf{Enc}, \mathsf{Dec})$, we may construct a ke-PKE scheme $(\mathsf{Gen}', \mathsf{Upd}, \mathsf{Enc}', \mathsf{Dec}')$ as follows.

- Algorithm $\mathsf{Gen}'(1^k, N)$ runs $\mathsf{Gen}(1^k, \ell)$, where $N \leq 2^{\ell+1} - 1$, 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 $SK_{w^{i+1}}$. If w^i is an internal node, compute $(SK_{w^i0}, SK_{w^i1}) \leftarrow \mathsf{Der}(PK, w^i, SK_{w^i})$ and push SK_{w^i1} and then SK_{w^i0} onto the stack. The new key on top of the stack is SK_{w^i0} (and indeed $w^{i+1} = w^i0$). In either case, node key SK_{w^i} is then erased.

- Algorithm $\mathsf{Enc}'(PK', i, M)$ runs $\mathsf{Enc}(PK, w^i, M)$. Note that w^i is publicly computable given i and N.
- Algorithm $\mathsf{Dec}'(PK', i, SK'_i, M)$ runs $\mathsf{Dec}(PK, w^i, SK_{w^i}, M)$. Note that SK_{w^i} is always 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 straightforward reduction. Assume we have an adversary A' with advantage $\varepsilon(k)$ in an fs-CPA (resp. fs-CCA) attack against $(\mathsf{Gen}', \mathsf{Upd}, \mathsf{Enc}', \mathsf{Dec}')$. We construct an adversary A that obtains advantage $\varepsilon(k)/N$ in the corresponding attack against the underlying BTE scheme $(\mathsf{Gen}, \mathsf{Der}, \mathsf{Enc}, \mathsf{Dec})$. Since N 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 1 that without loss of generality A' makes its $\mathsf{breakin}$ query before its $\mathsf{challenge}$ query), if $j \le i^*$ 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 $\varepsilon(k)/N$. ■

Analysis of complexity parameters. Each of the four operations (key generation, key update, encryption, and decryption) requires at most one operation of the underlying BTE scheme. We have also noted in the previous section how the secret keys corresponding to any time period can be stored using only $O(\log N)$ group elements (rather than the naïve $O(\log^2 N)$). 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 polylogarithmic 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 [31], we can further improve this scheme so that its efficiency depends only logarithmically on the number of time periods *elapsed thus far* (a simple pre-order traversal using a tree of depth $\omega(\log k)$ results in a scheme with superlogarithmic dependence on N for any $N = \mathsf{poly}(k)$).

Acknowledgments

The third author is very grateful to Craig Gentry for helpful discussions regarding [19] and for providing him with a preliminary version of that work.

References

[1] M. Abdalla and L. Reyzin. A new forward-secure digital signature scheme. *Asiacrypt '00*, LNCS vol. 1976, pp. 116–129, Springer-Verlag, 2000.

[2] A. Aho, J. Hopcroft, and J. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1975.

[3] R. Anderson. Two remarks on public key cryptology. Invited Lecture, *ACM-CCS '97*. `http://www.cl.cam.ac.uk/ftp/users/rja14/forwardsecure.pdf`.

[4] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In *Eurocrypt '92*, LNCS vol. 658, pp. 307–323, Springer-Verlag, 1992.

[5] D. Beaver. Plug-and-play encryption. *Crypto '97*, LNCS vol. 1294, pp. 75–89, Springer-Verlag, 1997.

[6] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. *Crypto '99*, LNCS vol. 1666, pp. 431–448, Springer-Verlag, 1999.

[7] M. Bellare and B. Yee. Forward security in private-key cryptography. *CT-RSA 2003*, to appear. Preliminary version at `http://eprint.iacr.org/2001/035/`.

[8] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. *Crypto '98*, LNCS Vol. 1462, pp. 26–45, Springer-Verlag, 1998.

[9] M. Bellare and M. Yung. Certifying permutations: non-interactive zero-knowledge based on any trapdoor permutation. *J. Cryptology* 9(3): 149–166 (1996).

[10] D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *Crypto '01*, LNCS vol. 2139, pp. 213–229, Springer-Verlag, 2001. Full version to appear in *SIAM J. Computing* and available at `http://eprint.iacr.org/2001/090`.

[11] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure computation. *STOC '96*, pp. 639–648, ACM, 1996. Also MIT-LCS-TR #682, 1996.

[12] I.B. Damgård. Collision free hash functions and public-key signature schemes. *Eurocrypt '87*, LNCS vol. 304, pp. 203–216, Springer-Verlag, 1987.

[13] I.B. Damgård and J.B. Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. *Crypto '00*, LNCS vol. 1880, pp. 432–450, Springer-Verlag, 2000.

[14] Y. Desmedt and Y. Frankel. Threshold cryptosystems. *Crypto '89*, LNCS vol. 435, pp. 307–315, Springer-Verlag, 1989.

[15] W. Diffie, P. C. Van-Oorschot, and M. J. Weiner. Authentication and authenticated key exchanges. *Designs, Codes, and Cryptography* 2:107–125, 1992.

[16] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong key-insulated signature schemes. *PKC '03*, LNCS vol. 2567, pp. 130–144, Springer-Verlag, 2003.

[17] U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero-knowledge proofs under general assumptions. *SIAM J. Computing* 29(1): 1–28 (1999).

[18] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Crypto '99*, LNCS 1666, pp. 537–554, Springer-Verlag, 1999.

[19] C. Gentry and A. Silverberg. Hierarchical identity-based cryptography. *Asiacrypt '02*, LNCS vol. 2501, pp. 548–566, Springer-Verlag, 2002.

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

[21] O. Goldreich. *Foundation of Cryptography, vol. 2*. Available on-line from `http://www.wisdom.weizmann.ac.il/~oded/foc-vol2.html`.

[22] C. G. Günther. An identity-based key-exchange protocol. *Eurocrypt '89*, LNCS vol. 434, pp. 29–37, Springer-Verlag, 1989.

[23] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, April 1988.

[24] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. *Eurocrypt '02*, LNCS vol. 2332, pp. 466–481, Springer-Verlag, 2002.

[25] G. Itkis and L. Reyzin. Forward-secure signatures with optimal signing and verifying. *Crypto '01*, LNCS vol. 2139, pp. 499–514, Springer-Verlag, 2001.

[26] 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/`.

[27] J. Katz and M. Yung. Complete characterization of security notions for probabilistic private-key encryption. *STOC '00*, ACM, 2000.

[28] A. Kozlov and L. Reyzin. Forward-secure signatures with fast key update. *Security in Communication Networks*, LNCS vol. 2576, pp. 247–262, Springer-Verlag, 2002.

[29] H. Krawczyk. Simple forward-secure signatures from any signature scheme. *ACM-CCS '00*, pp. 108–115, ACM, 2000.

[30] Y. Lindell. A simpler construction of CCA2-secure public-key encryption under general assumptions. *Eurocrypt '03*, to appear.

[31] T. Malkin, D. Micciancio, and S. K. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. *Eurocrypt '02*, LNCS vol. 2332, pp. 400–417, Springer-Verlag, 2002.

[32] M. Naor and M. Yung, Public key cryptosystems provably secure against chosen ciphertext attacks, *STOC '90*, pp. 427–437, ACM, 1990.

[33] J.B. Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. *Crypto '02*, LNCS vol. 2442, pp. 111-126, Springer-Verlag, 2002.

[34] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. *PODC '91*, pp. 51–59, ACM, 1991.

[35] C. Rackoff and D. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. *Crypto '91*, LNCS vol. 576, pp. 433–444, Springer-Verlag, 1991.

[36] A. Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. *FOCS '99*, pp. 543–553, IEEE, 1999.

[37] A. Shamir. How to share a secret. *Comm. of the ACM* 22(11):612–613, 1979.

A Basing HIBE on BTE

Here we show how one can construct a full-blown hierarchical identity-based encryption scheme (HIBE) from any binary-tree encryption scheme (BTE). The construction is straightforward: we encode all the identities in binary using a collision-resistant hash function; this maps "ID-vectors" to fixed-length strings. In this way, any "ID-vector" is mapped to a node in the complete binary tree of the underlying BTE scheme. Below, we define the HIBE functionality similarly to [24, 19],[5] introduce the notions of SN-security for HIBE, and then show that the above obvious construction indeed transforms any SN-secure BTE to a SN-secure HIBE.

A.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 $()$. If $v = (v_1, \ldots, v_\ell)$ is an ID-vector and $v_{\ell+1}$ is any string, then by $v.r$ 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 ℓ for the depth of the tree. It returns a master public key PK and an initial (root) secret key $SK_{()}$. (This algorithm is called Setup in [24] and Root-setup in [19].) We assume that the values of k and ℓ are implicit in PK and all node secret keys.

[5] We comment that our definition of HIBE differs slightly from the ones in [24, 19] in that it allows encrypting messages to the root of the tree (i.e., to the empty ID-vector). This difference is essentially just a matter of taste.

- 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 $v.r$. (This algorithm is called KeyGen$_i$ in [24] and Extraction in [19].)
- 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.

We make the standard correctness requirement: namely, for any $(PK, SK_{()})$ output by $\mathsf{Gen}(1^k, \ell)$, 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 $M = \mathsf{Dec}(PK, v, SK_v, \mathsf{Enc}(PK, v, M))$. ■

The notion of SN-security is a relaxation of the usual notion of security for HIBE schemes, requiring that the attacker commit to the ID-vector to be attacked before it sees the public key (similar to the SN-security notion of BTE scheme). A difference here is that one cannot give the adversary the secret keys of "all the siblings on the path to the target ID-vector" since there are too many of those. Instead, we allow the attacker to request the secret key corresponding to any ID-vector, except for prefixes of the target ID-vector.

Definition 8 A HIBE scheme is *secure against selective-node, chosen-plaintext attacks* (SN-CPA) if for all polynomially-bounded functions $\ell(\cdot)$, the advantage of any PPT adversary A in the following game is negligible in the security parameter:

1. The adversary $A(1^k, \ell(k))$ outputs an ID-vector $v^* \in (\{0,1\}^*)^{\leq \ell(k)}$.
2. Algorithm $\mathsf{Gen}(1^k, \ell(k))$ 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.[6]
4. The adversary generates a request $\mathsf{challenge}(M_0, M_1)$. A random bit b is selected and the adversary is given $C^* = \mathsf{Enc}(PK, v^*, M_b)$.
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. ■

A definition of security against chosen-ciphertext attacks is an obvious extension of the above.

Definition 9 A HIBE scheme is *secure against selective-node, chosen-ciphertext attacks* (SN-CCA) if for all polynomially-bounded functions $\ell(\cdot)$, the advantage of any PPT adversary A in the following game is negligible in the security parameter:

1. The adversary $A(1^k, \ell(k))$ outputs an ID-vector $v^* \in (\{0,1\}^*)^{\leq \ell(k)}$.
2. Algorithm $\mathsf{Gen}(1^k, \ell(k))$ outputs $(PK, SK_{()})$. The adversary is given PK.

[6]If the Ext algorithm is randomized, then we assume that all these queries are answered consistently. For example, if the adversary first asks for the secret key of $v.s$ and then the secret key of v, then the randomness in generating the secret key of v is the same as the randomness that was used during the generation of the secret key of $v.s$.

3. The adversary may ask for the secret key corresponding to any ID-vector v, as long as v is not a prefix of the target ID-vector v^*. Such query is answered with the secret key SK_v, correctly generated for v using the Ext algorithm.

   The adversary may also query a decryption oracle $\mathsf{Dec}^*(\cdot,\cdot)$. On query $\mathsf{Dec}^*(v, C)$ with $v \in (\{0,1\}^*)^{\leq \ell(k)}$, a key SK_v is derived from $SK_{()}$ (if one was not derived previously) and the adversary is given $M = \mathsf{Dec}(PK, v, SK_v, C)$.

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

5. The adversary can keep asking for secret keys as above, and may also continue to query $\mathsf{Dec}^*(\cdot,\cdot)$, except that it may not query $\mathsf{Dec}^*(v^*, C^*)$ (but it can query $\mathsf{Dec}^*(v, C^*)$ with $v \neq v^*$).

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. ■

A.2 From BTE to HIBE

We now show a simple transformation which converts an SN-secure BTE scheme to an SN-secure HIBE scheme. In this transformation we use collision-resistant hashing to map an ID-vector with a bounded number of entries to a bounded-length string. Namely, we apply the hash function separately to each entry in the vector, thus obtaining a string whose length depends only on the number of entries in the input ID-vector (and not the length of these entries).

Collision-resistant hashing. Recall that a collision-resistant hash function [12] consists of two algorithms: the seed-generation algorithm sGen that (given the security parameter) picks a seed for the function, and a hashing algorithm Hash that given a seed and an arbitrary-length input string, produces the fixed-length output. The function has the property that for any seed generated by $\mathsf{sGen}(1^k)$, the output length of the hashing algorithm is always equal to k. The collision-resistance property asserts that any poly-time adversary that is given a random seed s (generated by $\mathsf{sGen}(1^k)$), can only find strings $r_1 \neq r_2$ such that $\mathsf{Hash}(s, r_1) = \mathsf{Hash}(s, r_2)$ with probability negligible in k.

Below it will be convenient to refer to the entry-wise application of the hash function to ID-vectors. If s is a seed generated by $\mathsf{sGen}(1^k)$ and $v = (v_1, \ldots, v_\ell)$ is an ID-vector, then $w = \mathsf{Hash}(s, v)$ refers to the string $H(s, v_1)|\cdots|H(s, v_\ell)$.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 node w using some underlying BTE scheme. In more detail, given a collision intractable hash function ($\mathsf{sGen}, \mathsf{Hash}$) and a BTE scheme ($\mathsf{Gen}$, Der, Enc, Dec), we may construct a HIBE scheme (Gen', Ext, Enc', Dec') as follows.

- Algorithm $\mathsf{Gen}'(1^k, \ell)$ runs $\mathsf{Gen}(1^k, k\ell)$ (for a binary tree of depth $k\ell$), and obtains (PK, SK_ε). It also run $\mathsf{sGen}(1^k)$ and gets the seed s. The public key of the HIBE is set to $PK' = (s, PK)$ and the root secret key is $SK'_{()} = SK_\varepsilon$.

- Algorithm $\mathsf{Ext}(PK', v, SK'_v, r)$ sets $w = \mathsf{Hash}(s, v)$ and $w' = w'_1 \ldots w'_k = \mathsf{Hash}(s, r)$. (Recall that the secret key SK'_v is nothing more than the secret key SK_w for the BTE scheme.) For $i = 1..k$, algorithm Ext uses the algorithm Der to derive the BTE secret key $SK_{w\ w'_1..w'_i}$ from the BTE secret key $SK_{w\ w'_1..w'_{i-1}}$. 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 collision-resistant hash function 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 SN-CPA (resp. SN-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 a HIBE scheme just 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 can commit to its target node $w^* = \mathsf{Hash}(s, v^*)$. Then B can use its own queries to the BTE scheme to answer all of the queries that A makes to the HIBE scheme, with only two exceptions. One exception is when 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) is when A asks a decryption query $\mathsf{Dec}^*(v, C^*)$ where $v \neq v^*$ but $\mathsf{Hash}(s, v) = \mathsf{Hash}(s, v^*)$. It is easy to see that any of these exceptions imply finding collisions in the hash function. Hence, the probability of either of them occurring is negligible, and therefore B's advantage is only negligibly smaller than the advantage of A. ■

Theorems 2 and 5, along with the fact that the decisional BDH assumption implies the existence of collision-resistant hashing (since the decisional BDH assumption implies that the discrete logarithm problem in the relevant group is hard), we obtain:

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

B Basing NIZK on Publicly-Verifiable Trapdoor Predicates

We begin with a definition of publicly-verifiable trapdoor predicates that generalize the notion of trapdoor permutations. Somewhat informally, we replace the requirements that the domain is efficiently sampleable and that the permutation is efficiently computable by the (weaker) requirements that it is possible to efficiently sample uniform pairs $(x, \pi(x))$ and that given a pair (x, y) it is possible to efficiently determine whether or not $y = \pi(x)$. Then, we argue 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).

The formal definition we give here is patterned after the definition of trapdoor permutations (cf. [20, Definition 2.4.5]). Below we let $\bar{I} \subseteq \{0,1\}^*$ be an index set and denote $\bar{I}_k \stackrel{\text{def}}{=} \bar{I} \cap \{0,1\}^k$. For 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 tells us whether or not the pair (x, y) is of the form $(x, \pi(x))$.)

Definition 10 Let $\mathcal{F} = \{f_i : i \in \bar{I}\}$ be a collections of functions $f_i : D_i \times D_i \to \{0,1\}$ such that for all $i \in \bar{I}$ and $x \in D_i$, there is a unique y 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 $I(1^k) \in \bar{I}_k \times \{0,1\}^*$.
- *Uniform sampling of valid predicates*: For all $i \in \bar{I}$ we have

  – If $(x, y) \leftarrow D(i)$ then $f_i(x, y) = 1$.
  – For all $\tilde{x} \in D_i$, $\Pr[(x, y) \leftarrow D(i) : x = \tilde{x}] = 1/|D_i|$.

  (Note that this also guarantees that y is uniformly distributed in 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 probability is negligible in k:
$$\Pr[(i, td) \leftarrow I(1^k); (x, y) \leftarrow D(i) : A(i, y) = x].$$
- *Finding a valid "match" with trapdoor*: For all k, any pair (i, t) output by $I(1^k)$, and any $y \in D_i$ we have $f_i(F^{-1}(t, y), y) = 1$.

■

It is not hard to see that a BDH parameter generator $\mathcal{IG}$ satisfying the computational BDH assumption (cf. 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" (cf. [26]). Specifically, making the index a pair of random elements P, Q, we can define the predicate as $f_{P,Q}(R_1, R_2) = 1$ iff $\log_P R_2 = \log_Q R_1$. Now, verifying the equality is just an instance of the decisional Diffie-Hellman problem, while computing R_2 from P, Q, and R_1 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 $(\beta Q, \beta P)$.
- $F(i, R_1, R_2)$ (with $R_1, R_2 \in \mathbb{G}_1$) outputs 1 iff $\hat{e}(P, R_1) = \hat{e}(Q, R_2)$.
- $F^{-1}(\alpha, R_2)$ outputs αR_2.

It is not difficult to see (e.g., [20, Section 4.10.2]) that publicly-verifiable trapdoor predicates satisfying some additional assumptions are sufficient to implement the "hidden-bits paradigm" and hence are sufficient for general NIZK. These additional assumptions, informally, relate to:

1. The ability to efficiently recognize elements of the index set I, or to prove that a given i is indeed in I (cf. [9]).
2. The existence of a sampling algorithm D' which, on input $i \in I$ 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:
$$\Pr[(i, td) \leftarrow I(1^k); \omega \leftarrow \{0, 1\}^*; y \leftarrow D'(i; \omega) : A(i, y, \omega) = x].$$
   (Note that A is given the random coins of D'. See [21, Appendix C.1] 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. In particular, since the receiver — and not a third party — establishes the "common reference string", a number of simplifications are possible. Namely, the receiver simply generates parameters $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P)$ (note that, in our case, these parameters may simply be those used by the underlying encryption scheme itself) and publishes a sufficiently-long sequence $R_1, \ldots, R_\ell$ 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 for a publicly verifiable trapdoor predicate). Note that since the sender has the trapdoor α, he may indeed implement the "hidden-bits paradigm" using an appropriate hardcore bit for the trapdoor predicate thus defined.