Forward-Secure Hierarchical ID-Based Cryptography

July 29, 2003

Abstract

We present a forward-secure hierarchical identity-based encryption (FHIBE) scheme, which is based on the hierarchical identity-based encryption (HIBE) [16] scheme by Gentry and Silverberg. Canetti, Halevi and Katz presented a forward-secure public key encryption [11] scheme based on HIBE [16] scheme. They give the formal definition of Binary Encryption Tree (BET), which is a relaxed version of HIBE and is essential to their forward-secure encryption. We unify their idea with HIBE scheme, and present a forward-secure hierarchical identity-based encryption scheme. In the FHIBE scheme, secret keys of each entity on the hierarchy are updated at regular intervals throughout the lifetime of the system; furthermore, exposure of an entity's secret key corresponding to a given interval does not enable an adversary to break the ancestors of the entity for any prior time period. Entities can join in the hierarchy at any time and at any position, and are able to update their secret keys on their own once they are initialized by their parent entities. These features are important in the distributed settings.

The forward-secure hierarchical identity-based encryption scheme can be generalized into a collusion resistant multiple hierarchical identity-based encryption (MHIBE) scheme, where a message can be encrypted under multiple identities of a user.

Keywords: Bilinear Diffie-Hellman, Encryption, forward security, identity-based encryption, multiple hierarchies.

1 Introduction

1.1 Hierarchical Identity-based Encryption

In 1984, Shamir [27] proposed an idea of identity-based cryptosystem in which the public key can be an arbitrary string. The main advantage of this scheme is to largely reduce the need for public key certificates and certificate authorities. When Alice wants to send mail to Bob, she merely encrypts her message using Bob's public key that is derived from his identifying information. When Bob receives the encrypted mail he contacts a third party, which we call Private Key Generator (PKG). Bob authenticates himself to the PKG in the same way he would authenticate himself to a CA and obtains his private key from PKG. Bob can decrypt the message. A fully functional and efficient scheme of identity-based encryption (BF-IBE) was not found until recently by Boneh and Franklin [8]. The security of their scheme bases on Bilinear Diffie-Hellman Problem [8].

Horwitz and Lynn [19] introduced hierarchical identity-based encryption (HIBE) scheme, and proposed a 2-level HIBE scheme with total collusion-resistance at the first level and with partial collusion-resistance at the second level. Gentry and Silverberg [16] extended BF-IBE scheme and presented a fully scalable hierarchical identity-based encryption scheme, which has total collusion resistance on any arbitrary number of levels. HIBE scheme, which aimed to alleviate the workload of the PKG, allows a root PKG to distribute the workload by delegating private key generation and identity authentication to lower-level PKGs. In HIBE scheme, a root PKG needs only generate private keys for domain-level PKGs, who in turn generate private keys for users in their domains in the next level. Bob only has to authenticate himself to the domain-level PKG that he directly belongs to in order to obtain his private key. To encrypt a message, Alice needs only obtain the public parameters of Bob's root PKG (and Bob's identifying information); there are no lower-level parameters.

1.2 Forward security

A spectrum of methods have been introduced to deal with the threat of exposure of secret keys, including secret sharing [26], threshold cryptography [12, 23, 20], and proactive cryptography [25, 18]. In this paper we study one such technique: forward-secure cryptography. The notion of forward secrecy was first proposed in the context of key exchange protocols by Günther [15] and later by Diffie *et al.* [13]. The basic idea is that compromise of long-term keys does not compromise past session keys. A forward-secure key exchange protocol naturally gives rise to a forward-secure interactive encryption scheme [11]. The notion of non-interactive forward security was proposed by Anderson [2] in 1997 and later formalized by Bellare and Miner [4]. In this model, secret keys are updated at regular intervals throughout the lifetime of the system; furthermore, exposure of a secret key corresponding to a given interval does not enable an adversary to break the system (in the appropriate sense) for any prior time period. The model inherently cannot prevent the adversary from breaking the security of the system for any subsequent time period.

Bellare and Miner [4] formalized and provided the first construction to a forward-secure digital signature scheme suggested by Anderson [2]. Following their scheme there is a sequence of constructions of improved forward-secure signature schemes [3, 1, 24]. Bellare and Yee [6] provided a comprehensive treatment of forward-security in the context of private key based cryptographic primitives.

The first forward-secure public key encryption scheme is given by Canetti, Halevi and Katz

[11]. Their scheme is based on bilinear Diffie-Hellman assumption [8] in the random oracle model [5]. They rigorously defined the security of a forward-secure public key encryption scheme and also gave efficient constructions based on the Gentry and Silverberg [16] construction of a hierarchical identity-based encryption scheme [16, 19].

1.3 Motivation for Forward-secure Hierarchical Identity-based Encryption Scheme

The standard notion of IBE or HIBE security is extremely vulnerable to leakage of secret keys, which, over the lifetime of the scheme, may be quite a realistic threat. Because the public keys are not ID-based, the forward-secure public key encryption scheme by Canetti, Halevi and Katz [11] cannot be directly applied to (hierarchical) identity-based encryption schemes. A forward-secure hierarchical identity-based encryption would be desirable.

Canetti and Halevi proposed informally a simple forward-secure hierarchical identity-based encryption [10] by appending all the time periods as identities after the user's ID-tuple. During the t-th epoch use the identity of the t-th node on that chain. The key-update operation consists of introducing a new node (with some fixed identity, e.g., the letter **I**), computing its personal secret key, and erasing the personal secret key of the previous node. To encrypt during the t-th epoch, one uses the master public key, together with the chain of t identities **I.I.I ... I**. This scheme incurs no extra storage requirements for secret keys compared to our scheme. However, the size of public key increases significantly as time goes on and so does the length of ciphertext and the size of secret keys. Furthermore there is a serious scalability issue with this scheme. An entity extends its ID-tuple by introducing **I** nodes and erases its original secret key (secret key at time 0). This makes it impossible for the entity to create any immediate child nodes from its original ID-tuple (the ID-tuple at time 0) in the future, because it lost the original secret key. Therefore the hierarchy structure of PKGs and users has to be static. The only way to add a child node in this scheme is when the system is first set up, which means the whole hierarchy has to be in place when the root PKG is set up and is therefore unable to handle dynamic joins.

1.4 Our Contribution

We unify forward-secure public key encryption [11] scheme with HIBE [16] scheme by Gentry and Silverberg and present a forward-secure HIBE (FHIBE) scheme. We give the formal definitions and a practical construction of the FHIBE scheme. The secret keys in an FHIBE scheme are forward-secure by evolving with time, while users' public information is kept the same. Entities can join the hierarchy at any time and are able to update their secret keys independently, once they are initialized by their parent entities. The amount of communication among users and their parent PKG is minimized in order to reduce overhead and security vulnerability. These features are very important in distributed and dynamic settings. The scheme has total collusion resistance and chosen ciphertext security in the random oracle model [5], regardless of the level of the ID-tuple used for encryption and the time period, assuming the difficulty of the BDH problem [7, 16, 11].

We generalize the forward-secure hierarchical identity-based encryption (FHIBE) scheme into a collusion resistant multiple hierarchical identity-based encryption (MHIBE) scheme. In a large organization a user may own multiple identities, each of which is represented by an ID-tuple. An MHIBE scheme is useful when a message needs to be encrypted under multiple identities to ensure only the one with all the identities can decrypt it. An encrypted message should not be decrypted even if some other users colluding with each other. We note that the FHIBE scheme is a special

case of our MHIBE scheme, in that in FHIBE scheme, time can be viewed as another identity of a user, the same way as the user's original ID-tuple. Therefore the identities in MHIBE scheme capture a broad sense of meaning. For example, an identity may be the user's geographic location, network address, or the current time, as in the case of FHIBE scheme. One restriction of our MHIBE scheme is that the ID-tuples that a user owns have to share the same root PKG. Smart [28] presents a mechanism to encrypt to an arbitrary collection of non-hierarchical identities using a variant of the Boneh-Franklin identity-based encryption [8] scheme for access control purposes.

The details of the dependencies of various parameters on total number of time periods N and the length of an ID-tuple L, and the number of identities that a user owns in the MHIBE scheme M are summarized in Table 1. Key generation time includes the root PKG setup time.

Parameters	FHIBE	MHIBE
Key generation time	$\mathcal{O}(L\log(N))$	$\mathcal{O}(L^M)$
Encryption/Decryption time	$\mathcal{O}(L\log(N))$	$\mathcal{O}(L^M)$
Key update time	$\mathcal{O}(\log(N))$	$\mathcal{O}(L^{M-1}\log(N))^*$
Ciphertext length	$\mathcal{O}(L\log(N))$	$\mathcal{O}(L^M)$
Public key size	$\mathcal{O}(\log(N)+L)$	$\mathcal{O}(LM)$
Secret key size	$\mathcal{O}(L\log^2(N))$	$\mathcal{O}(L^M)$

Table 1: Dependency of parameters on the total number of time periods N, the length of an ID-tuple L, and the number of ID-tuples a user has in the FHIBE and MHIBE scheme. * This only applies when one of the identity in MHIBE scheme is time.

The paper is organized as follows. In Section 2 we give the definitions for FHIBE scheme and its security. In Section 3 we recall the bilinear Diffie-Hellman assumption [8] and give the construction of a FHIBE scheme and the theorem about the security. In Section 4 we describe a multiple hierarchical identity-based encryption scheme, which is a generalization from our FHIBE scheme.

2 Definitions

2.1 Overview

This section defines the forward-secure hierarchical identity-based encryption scheme, and related security. We also give the definitions for bilinear Diffie-Hellman assumption as in [7, 16, 11]. Before we present the formal definitions, we first give an informal discussion over the FHIBE scheme.

In the FHIBE scheme, the secret keys associated with an ID-tuple evolve with time. At any time period i an entity joins in the system (hierarchy), its parent node computes its decryption key corresponding to time period i and other values necessary for the entity to compute its own future secret keys. Once the newly joining entity receives this secret information, at the end of each period it periodically updates its secret key based on the current one, and discards the current one. During time period i, a message is encrypted under an ID-tuple and the time period i. Decryption of a ciphertext requires the secret key of the ID-tuple corresponding to time period i.

Our definitions of the FHIBE scheme make use of the definitions of binary tree encryption (BTE) scheme and forward-secure public key encryption [11] scheme. BTE scheme is a relaxed version of the HIBE [16] scheme. As in HIBE scheme, in BTE scheme each node in the tree has a

corresponding secret key, which can be used to derive the secret keys for the children of that node. The only difference between BTE and HIBE is that in BTE the tree is a binary tree, where the children of a node w are labeled $w0$ and $w1$, whereas in HIBE the tree can have arbitrary degree, and a child of node can be labeled for any arbitrary string.

Based on their definitions for BTE scheme, Canetti *et al.* give the definition of a key-evolving public key encryption (ke-PKE) scheme, which includes four algorithms: GENERATE, UPDATE, ENCRYPT, and DECRYPT. The initial key SK_0 is generated along with a public key by algorithm GENERATE. At time i, UPDATE takes in key SK_i and returns a new key SK_{i+1} for time period $i+1$.

Although it is natural to define forward-secure public key encryption scheme based on HIBE [11] scheme, due to the distributed nature of the hierarchy and involvement of time, the definitions of an FHIBE scheme has to take into consideration the dynamic requirement of practical applications. An FHIBE scheme has to be dynamically scalable and to allow new entities to join at any time after the root PKG is set up. In other words, the hierarchy structures may dynamically change with time. To minimize communication among the entities, a feasible FHIBE scheme should allow users, after they are set up with the help of their parents, to update their secret keys on their own without any assistance from the parent nodes.

We present our definitions for an FHIBE scheme, based on HIBE [16] scheme and the construction of forward-secure public key encryption [11] scheme with logarithmic complexity. The FHIBE scheme is described by defining six algorithms: ROOT SETUP, EXTRACT, LOWER-LEVEL SETUP, UPDATE, ENCRYPT, and DECRYPT. We define the security of FHIBE similar to [7, 16, 11]. Because we build the forward-security based on HIBE scheme, we are able to define the security of FHIBE very similarly to HIBE scheme.

2.2 Scheme Definitions

Notations: **Standard GMR notation:** Let $A(\cdot)$ be an algorithm. $y \leftarrow A(x)$ denotes that y was obtained by running A on input x. In case A is deterministic, then this y is unique; if A is probabilistic, then y is a random variable. Let b be a boolean function. The notation $(y \leftarrow A(x) \ : \ b(y))$ denotes the event that $b(y)$ is true after y was generated running A on input x. Finally, the statement such as $\Pr[y \leftarrow A(x); z \leftarrow B(y) \ : \ b(z)] = \alpha$ means that the probability that $b(z)$ is TRUE after the value z was obtained by first obtaining y by running algorithm A on input x, and then running algorithm B on input y.

Concatenation: A $\circ$ B denotes string A concatenated with string B. **Keys:** There are two types of keys: $sk_{w,(\mathrm{ID}_0,\ldots,\mathrm{ID}_h)}$ and $SK_{i,(\mathrm{ID}_0,\ldots,\mathrm{ID}_h)}$. $sk_{w,(\mathrm{ID}_0,\ldots,\mathrm{ID}_h)}$, the node key, is the key associated with some prefix of the bit representation of time i and an ID-tuple $(\mathrm{ID}_0,\ldots,\mathrm{ID}_h)$. Key $SK_{i,(\mathrm{ID}_0,\ldots,\mathrm{ID}_h)}$ denotes the key associated with time i and an ID-tuple $(\mathrm{ID}_0,\ldots,\mathrm{ID}_h)$. When this causes no confusion, we denote the keys as $sk_{w,h}$ and $SK_{i,h}$, respectively.

Time Period: As usual in forward-secure public key encryption [11] scheme, we assume for simplicity that the total number of time periods N is a power of 2; that is $N = 2^t$.

ID-tuple: As in HIBE scheme [16], an entity has a position in the hierarchy, defined by its tuple of IDs: $(\mathrm{ID}_0,\ldots,\mathrm{ID}_h)$. The entity's ancestors in the hierarchy are the users / PKGs whose ID-tuple are $\{(\mathrm{ID}_0,\ldots,\mathrm{ID}_i) : 0 \le i < h\}$. We let the root PKG have an empty string denoted as ID_0 for its ID. The ID-tuples of lower-level PKGs or users may be denoted without ID_0 as $(\mathrm{ID}_1,\ldots,\mathrm{ID}_h)$.

Forward-secure Hierarchical Identity-based Encryption (FHIBE) scheme: an FHIBE

scheme is specified by six algorithms: ROOT SETUP, EXTRACT, LOWER-LEVEL SETUP, UPDATE, ENCRYPT, AND DECRYPT:

Root Setup: The root PKG takes a security parameter K and the total number of time periods N, and returns *params* (system parameters) and the initial root key $SK_{0,0}$. The system parameters include a description of the message space $\mathcal{M}$ and the ciphertext space $\mathcal{C}$. The system parameters will be publicly available, while only the root PKG knows the initial root key.

Extract: An entity (a PKG or a user) with ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$ uses the value $sk_{w,h}$ associated with w to compute values $sk_{(w0),h}$ and $sk_{(w1),h}$ for the two child nodes of w.

Lower-level Setup: During a time period i, a lower-level entity (user or lower-level PKG) joins in the system at level h. Its parent at level $h-1$ sets the entity's key $SK_{i,h}$ associated with time period i.

Update: During the time period i, a PKG (whether the root one or a lower-level one) with ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$ uses $SK_{i,h}$ to compute his key $SK_{(i+1),h}$ for the next time period, and erases $SK_{i,h}$.

Encrypt: A sender inputs *params*, the index i of the current time period, $M \in \mathcal{M}$ and the ID-tuple of the intended message recipient, and computes a ciphertext $C \in \mathcal{C}$.

Decrypt: During the time period i the user with the ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$ inputs *params*, $C \in \mathcal{C}$, and its key $SK_{i,h}$ associated with time period i and the ID-tuple, and returns the message $M \in \mathcal{M}$.

Encryption and decryption must satisfy the standard consistency constraint, namely when SK_{ih} is the secret key generated by algorithm EXTRACT for ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$ and time period i, then: $\forall M \in \mathcal{M} : \textsc{Decrypt}(params, SK_{i,h}, C) = M$,where $C = \textsc{Encrypt}(params, i, (\mathrm{ID}_0, \ldots, \mathrm{ID}_h), M)$.

2.3 Security Definitions

As in IBE [7] and HIBE [17] schemes, we allow an attacker to make *lower-level setup queries*. Also, we allow the adversary to choose the time period and identity on which it wishes to be challenged. Notice that an adversary may choose the time period and identities of its targets adaptively or nonadaptively. An adversary that chooses its targets adaptively will first make hash queries and lower-level setup queries, and then choose its targets based on the results of these queries. A nonadaptive adversary, on the other hand, chooses its targets independently from the results of the hash queries and lower-level setup queries he makes. Security against an adaptive-chosen-target adversary is the stronger notion of security, which is captured below.

Chosen-ciphertext Security:

We say an FHIBE scheme is semantically secure against adaptive chosen ciphertext, adaptively chosen time period and targets chosen target attack, if no polynomial time bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following game.

Setup: The challenger takes a security parameter k, and runs the ROOT SETUP algorithm. It gives the adversary the resulting system parameters *params*. It keeps the root secrets to itself.

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

1. Public key query ($\mathrm{time}_i \circ \mathrm{ID\text{-}tuple}_i$): challenger runs a hash algorithm on ($\mathrm{time}_i \circ \mathrm{ID\text{-}tuple}_i$) to obtain the public key $H_1(\mathrm{time}_i \circ \mathrm{ID\text{-}tuple}_i)$ corresponding to ($\mathrm{time}_i \circ \mathrm{ID\text{-}tuple}_i$).
2. Lower-level setup query ($\mathrm{time}_i \circ \mathrm{ID\text{-}tuple}_i$): The challenger runs the LOWER-LEVEL SETUP algorithm to generate the private key $SK_{(\mathrm{time}_i,\ \mathrm{ID\text{-}tuple}_i)}$ corresponding to ($\mathrm{time}_i \circ \mathrm{ID\text{-}tuple}_i$), and sends $SK_{(\mathrm{time}_i, \mathrm{ID\text{-}tuple}_i)}$ to the adversary.

3. Decryption query $(\text{time}_i, \text{ID-tuple}_i, C_i)$: the challenger runs the LOWER-LEVEL SETUP algorithm to generate the private key $SK_{(\text{time}_i,\text{ID-tuple}_i)}$ corresponding to $(\text{time}_i \circ \text{ID-tuple}_i)$, and runs the Decryption algorithm to decrypt C_i using $SK_{(\text{time}_i,\text{ID-tuple}_i)}$, and sends the resulting plaintext to adversary.

These queries may be asked adaptively. Also, the queried ID-tuple_i may corresponding to a position at any level in the ID hierarchy.

Challenge: Once the adversary decides that phase 1 is over, it outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$, and a time period t and an ID-tuple on which it wishes to be challenged. The constraint is that no lower-level setup query has been issued for the ID-tuple or any of its ancestors for any time $t_i \leq t$.

The challenger picks a random bit $b \in \{0,1\}$, and set $C = \text{ENCRYPT}(params, t, \text{ID-tuple}, M_b)$. It sends C as a challenge to the adversary.

Phase 2: the adversary issues more queries $q_{m+1}, \ldots, q_n$, where q_i is one of:

1. Public key query $(\text{time}_i \circ \text{ID-tuple}_i)$: the challenger responds as in **Phase 1**.
2. Lower-level setup query $(\text{time}_i \circ \text{ID-tuple}_i)$, where the time period and ID-tuple are under the same restriction as in **Challenge**: the challenger responds as in **Phase 1**.
3. Decryption query $(\text{time}_i \circ \text{ID-tuple}_i, C_i) \neq (t \circ \text{target ID-tuple}, C)$: the challenger runs algorithm EXTRACT to generate the private key $SK_{(\text{time}_i,\text{ID-tuple}_i)}$ corresponding to $(\text{time}_i \circ \text{ID-tuple}_i)$, runs the Decryption algorithm to decrypt C_i using $SK_{(\text{time}_i,\text{ID-tuple}_i)}$, and sends the resulting plaintext to adversary.

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

3 Forward-secure Hierarchical Identity-based Encryption (FHIBE) Scheme

We present a forward-secure hierarchical identity-based encryption scheme with one-way security in a format similar to IBE [7], HIBE [17] and forward-secure public key encryption [22] schemes. Using Fujisaki-Okamoto padding [14], the scheme below can be converted into one with chosen ciphertext security, as in IBE [7] and HIBE [16] schemes. The proof of security of our FHIBE scheme is shown in the Appendix.

3.1 Assumptions

The security of our FHIBE scheme is based on the difficulty of Bilinear Diffie-Hellman (BDH) problem [8]. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of some large prime order q. We write $\mathbb{G}_1$ additively and $\mathbb{G}_2$ multiplicatively. Our schemes make use of a bilinear pairing.

Admissible pairings: Following Boneh and Franklin [8], we call $\hat{e}$ an admissible pairing if $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_2$ is a map with the following properties:

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

We refer the readers to papers by Boneh and Franklin [8], Joux [21] and Boneh and Silverberg [9] for examples and discussions of groups that admit such pairings.

Bilinear Diffie-Hellman (BDH) Parameter Generator: As in IBE [7] scheme, a randomized algorithm $\mathcal{IG}$ is a BDH parameter generator if $\mathcal{IG}$ takes a security parameter $K > 0$, runs in time polynomial in K, and outputs the description of two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of the same prime order q and the description of an admission paring $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

BDH Problem: As in IBE [7] scheme, given a randomly chosen $P \in \mathbb{G}_1$, as well as aP, bP, and cP (for unknown randomly chosen $a, b, c \in \mathbb{Z}_q$), compute $\hat{e}(P, P)^{abc}$.

For the BDH problem to be hard, $\mathbb{G}_1$ and $\mathbb{G}_2$ must be chosen so that there is no known algorithm for efficiently solving Diffie-Hellman problem in either $\mathbb{G}_1$ or $\mathbb{G}_2$. Note that if the BDH problem is hard for a paring $\hat{e}$, then it follows that $\hat{e}$ is non-degenerate.

BDH Assumption: As in IBE [7] scheme, we say a BDH parameter generator $\mathcal{IG}$ satisfies the BDH assumption if the following is negligible in K for all PPT algorithm $\mathcal{A}$:

$$\Pr[(\mathbb{G}_1, \mathbb{G}_2, \hat{e}) \leftarrow \mathcal{IG}(1^K); P \leftarrow \mathbb{G}_1; a, b, c \leftarrow \mathbb{Z}_q : \mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP, cP) = \hat{e}(P, P)^{abc}].$$

3.2 One-way Identity-based Encryption

As in IBE [7], HIBE [16] schemes, the proof of security of our FHIBE system makes use of a weaker notion of security, which is one-way encryption (OWE). One-way security differs from chosen-ciphertext security in that decryption queries are not allowed and the challenge is guessing the message, as opposed to distinguishing ciphertexts of two messages. We say an FHIBE scheme is one-way if no polynomial time adversary has a non-negligible advantage against the challenger in the following game. (Phase 1 is omitted for nonadaptive adversary.)

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

Phase 1: The adversary makes public-key and/or lower-level setup queries as in Phase 1 above.

Challenge: Once the adversary decides that Phase 1 is over, it outputs a new time period t and a new ID-tuple on which it wishes to be challenged. The challenger picks a random $M \in \mathcal{M}$ and sets $C = \text{ENCRYPT}(params, t, \text{ID-tuple}, M)$. It sends C as a challenge to the adversary.

Phase 2: The adversary issues more public-key queries and more lower-level setup queries. The restrictions are the same as in the lower-level setup queries at Phase 2 of chosen-ciphertext security. The challenger responds as in Phase 1.

Guess: The adversary outputs a guess $M' \in \mathcal{M}$. The adversary wins the game if $M = M'$. We define the adversary's advantage in attacking the scheme to be $\Pr[M = M']$.

3.3 Construction of FHIBE scheme

Let $\mathcal{IG}$ be a BDH parameter generator for which the BDH assumption holds. The following gives a construction for a forward secure hierarchical identity-based public key encryption for N time periods.

As in the forward-secure public key encryption [11] scheme, we construct a full binary tree of height t. The root of this tree is labelled ϵ; all other nodes are recursively labelled as follows: if the label of a node is w, then its left child is labelled $w0$, and its right child is labelled $w1$. This way, for each $i \in \{0, N-1\}$, there is a leaf labelled with the binary representation of i. As in

forward-secure public key encryption scheme [11], we denote the i-bit prefix of a word $w_1 w_2 \ldots w_l$ by $w|_i$, that is, $w|_i = w_1 \ldots w_i$. Let $w|_0 = \epsilon$ and $w = w|_l$.

At the beginning of time, public parameters are generated. At time t, the person at level h with ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$ holds a secret key $SK_{t,(\mathrm{ID}_0,\ldots,\mathrm{ID}_h)}$. Recall that we denote key $SK_{t,(\mathrm{ID}_0,\ldots,\mathrm{ID}_h)}$ as $SK_{t,h}$ when this causes no confusion. $SK_{t,h}$ consists of $(sk_{t,h}, \{sk_{w,h}\})$, where w are all the labels of the right sibling, if one exists, of each ancestor of the node labelled t in the complete binary tree. $sk_{w,h}$ are also called node keys. At the beginning of time, as public parameters are created, the value $sk_{0,0}$ is created by the root PKG. Each $sk_{w,h}$, where w is any string, can be used to create an $sk_{w,u}$, where $u > h$, for ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_u)$, who is a child of ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$. The algorithm for doing so is LOWER-LEVEL SETUP.

Each $sk_{w,h}$, where w is any string, is also used to compute the value $sk_{(w \circ b),h}$, where $b = 0$ or 1. They are the child nodes of w on the binary tree. The algorithm for doing so is EXTRACT.

We must delete all information from which $sk_{t',h}$ for $t' < t$ can be inferred. The algorithm for computing all the right keys for the next time period and deleting all the old keys is UPDATE.

The public parameters, t, and $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$ are all that a sender needs to send an encrypted message to ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$ at time t using algorithm ENCRYPT.

The value $sk_{t,h}$ is all that the user with ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$ needs to decrypt at time t. The algorithm for doing so is DECRYPT.

Now, let us look at the contents of each $sk_{w,h}$ more closely. It has two components $S_{w,h}$ and $\mathcal{Q}_{w,h}$. $S_{w,h}$ is a point in $\mathbb{G}_1$, and $\mathcal{Q}_{w,h}$ contains a set of Q-values, which will be explained below. If w represents a leaf on the binary tree, $S_{w,h}$ and the Q-values in $\mathcal{Q}_{w,h}$ together are used for decryption for ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$ at time w. If w is an internal node on the binary tree, these values are used for generating future decryption keys.

Computing $S_{w,h}$ makes use of a series of secret $s_{w_1 \ldots w_k,(\mathrm{ID}_0,\ldots,\mathrm{ID}_j)}$ values, where $1 \leq k \leq |w|$ and $0 \leq j \leq h$. The shorthand notation of $s_{w_1 \ldots w_k,(\mathrm{ID}_0,\ldots,\mathrm{ID}_j)}$ is $s_{k,j}$, because $w_1 \ldots w_k$ and $(\mathrm{ID}_0, \ldots, \mathrm{ID}_j)$ should be clear from the context. Given a node w on the binary tree and an ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$, there is a secret s value for every (time, ID-tuple) combination, where time corresponds to a node that has some prefix of w including w itself and excluding the root node ϵ, and ID-tuple is some ancestor of ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$ including itself. Each $s_{k,j}$ is chosen randomly from $\mathbb{Z}_q$.

Each $s_{k,j}$ is also used for computing $Q_{k,j}$, which is called a Q-value. For each $s_{k,j}$ there is a Q-value $Q_{k,j}$, which is an element in $\mathbb{G}_1$. The Q-values are used in DECRYPT. $\mathcal{Q}_{w,h}$ is the union of some Q-values. All $s_{k,j}$ values are erased once $S_{w,h}$ and Q-values are computed. The construction is shown below.

ROOT SETUP$(1^k, N = 2^t)$: The root PKG does the following:

1. $\mathcal{IG}$ is run to generate groups $\mathbb{G}_1, \mathbb{G}_2$ of order q and bilinear map $\hat{e}$.
2. A random generator $P \leftarrow \mathbb{G}_1$ is selected along with random $s_\epsilon \leftarrow \mathbb{Z}_q$. Set $Q = s_\epsilon P$.
3. Choose a cryptographic hash function $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$. Choose a cryptographic hash function $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^n$ for some n. The security analysis will treat H_1 and H_2 as random oracles [5]. The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{G}_1^{(h+1)\times t} \times \{0,1\}^n$ where h is the level of the recipient. The system parameters are $params = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, H_1, H_2)$. The master key is $s_\epsilon \in \mathbb{Z}_q$. s_ϵ is also referred as $s_{1,0}$ below.

   The root PKG needs to generate value $sk_{w,0}$, where w are the current time period 0, and the internal nodes on the BTE tree whose bit representations are all 0 but the last bits. The sk

value for time 0 is denoted as $sk_{0^t,0}$. The other sk values are used for generating keys for future time periods and are denoted as $\{sk_{1,0}, sk_{(01),0}, \ldots, sk_{(0^{t-1}1),0}\}$.

The values are generated recursively using as follows.

(a) Set the secret point $S_{0,0}$ to $s_\epsilon H_1(0 \circ \mathrm{ID}_0) = s_\epsilon H_1(0)$, and $S_{1,0}$ to $s_\epsilon H_1(1)$. Recall ID_0, the ID of the root PKG, is defined as an empty string.

(b) Set the secret key $sk_{0,0} = (S_{0,0}, \emptyset)$ and $sk_{1,0} = (S_{1,0}, \emptyset)$. The root PKG uses $sk_{0,0}$ to recursively call algorithm EXTRACT (defined below) to generate his secret keys. Let $(sk_{w00,0}, sk_{w01,0}) = \textsc{Extract}(sk_{w0,0}, w0, \mathrm{ID}_0)$, for all $1 \leq |w0| \leq t-1$.

(c) Set the root PKG's key for time period 0 as $SK_{0,0} = (sk_{0^t,0}, \{sk_{1,0}, sk_{(01),0}, \ldots, sk_{(0^{t-1}1),0}\})$, and erases all other information.

$\textsc{Extract}(sk_{w,h}, w, \mathrm{ID}_0 \ldots \mathrm{ID}_h)$ outputs values $sk_{(w0),h}$, $sk_{(w1),h}$ for time nodes $w0$ and $w1$ of $(\mathrm{ID}_0 \ldots \mathrm{ID}_h)$.

1. Parse w as $w_1 \ldots w_l$ where $|w| = l$. Parse ID-tuple as $\mathrm{ID}_0, \ldots, \mathrm{ID}_h$. Parse $sk_{w,h}$ associated with time node w as $(S_{w,h}, \mathcal{Q}_{w,h})$, where $S_{w,h} \in \mathbb{G}_1$ and $\mathcal{Q}_{w,h} = \{Q_{k,j}\}$ for all $1 \leq k \leq l$ and $0 \leq j \leq h$, except $k = 1$ and $j = 0$.
2. Choose random $s_{(l+1),j} \in \mathbb{Z}_q$ for all $0 \leq j \leq h$.
3. Set $S_{(w0),h} = S_{w,h} + \sum_{j=0}^{h} s_{(l+1),j} H_1(w0 \circ \mathrm{ID}_0 \ldots \mathrm{ID}_j)$.
4. Set $S_{(w1),h} = S_{w,h} + \sum_{j=0}^{h} s_{(l+1),j} H_1(w1 \circ \mathrm{ID}_0 \ldots \mathrm{ID}_j)$.
5. Set $Q_{(l+1),j} = s_{(l+1),j} P$ for all $j \in \{0, h\}$.
6. Set $\mathcal{Q}_{(w0),h}$ and $\mathcal{Q}_{(w1),h}$ to be the union of $\mathcal{Q}_{w,h}$ and $Q_{(l+1),j}$ for all $1 \leq j \leq h$.
7. Output $sk_{(w0),h} = (S_{(w0),h}, \mathcal{Q}_{(w0),h})$ and $sk_{(w1),h} = (S_{(w1),h}, \mathcal{Q}_{(w1),h})$.
8. Erase $s_{(l+1),j}$ for all $0 \leq j \leq h$.

LOWER-LEVEL $\textsc{Setup}(SK_{i,(h-1)}, i, \mathrm{ID}_1 \ldots \mathrm{ID}_h)$: Let E_h be the entity that joins the hierarchy during the time period $i < N-1$ with ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_h)$. E_h's parent generates E_h's key $SK_{i,h}$ using his key $SK_{i,(h-1)}$ as follows:

1. Parse i as $i_1 \ldots i_t$. Recall $SK_{i,(h-1)}$ as $(sk_{i,(h-1)}, \{sk_{(i|_{k-1}1),(h-1)}\}_{i_k=0})$.
2. For each value $sk_{w,(h-1)}$ in $SK_{i,(h-1)}$, E_h's parent does the following to generate E_h's secret key $sk_{w,h}$:

   (a) Parse w as $w_1 \ldots w_l$, where $l \leq t$, and parse the secret key $sk_{w,(h-1)}$ as $(S_{w,(h-1)}, \mathcal{Q}_{w,(h-1)})$.

   (b) Choose random $s_{k,h} \in \mathbb{Z}_q$ for all $1 \leq k \leq l$. Recall $s_{k,j}$ is a shorthand for $s_{w_1 \ldots w_k, (\mathrm{ID}_0 \ldots \mathrm{ID}_j)}$, for time node $w_1 \ldots w_k$ and ID-tuple $(\mathrm{ID}_0 \ldots \mathrm{ID}_j)$.

   (c) Set the child entity E_h's secret point $S_{w,h} = S_{w,(h-1)} + \sum_{k=1}^{l} s_{k,h} H_1(w|_k \circ \mathrm{ID}_1 \ldots \mathrm{ID}_h)$.

   (d) Set $Q_{k,h} = s_{k,h} P$ for all $1 \leq k \leq l$. Let $\mathcal{Q}_{w,h}$ to be the union of $\mathcal{Q}_{w,(h-1)}$ and $Q_{k,h}$ for all $1 \leq k \leq l$.

   (e) Set $sk_{w,h}$ to be $(S_{w,h}, \mathcal{Q}_{w,h})$.
3. E_h's parent sets E_h 's $SK_{i,h} = (sk_{i,h}, \{sk_{(i|_k 1),h}\}_{i_k=0})$, and erases all other information.

UPDATE$(SK_{i,h}, i+1, (\mathrm{ID}_0 \ldots \mathrm{ID}_h))$ (where $i < N-1$): At the end of time i, entity with ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$ does the following to compute its private key for time $i+1$, as in [22].

1. Parse i as $i_1 \ldots i_t$ where $|i| = t$. Let $SK_{i,h}$ as $(sk_{(i|_t),h}, \{sk_{(i|_k 1),h}\}_{i_k=0})$. Erase $sk_{i|_t,h}$.
2. We distinguish two cases. If $i_t = 0$, simply output the remaining keys as the key $SK_{(i+1),h}$ for the next period for ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$. Otherwise, let $\tilde{k}$ be the largest value such that $i_{\tilde{k}} = 0$ (such $\tilde{k}$ must exist since $i < N-1$). let $i' = i_1 \ldots i_{\tilde{k}-1} 1$. Using $sk_{i',h}$ (which is included as part of $SK_{i,h}$), recursively apply algorithm EXTRACT to generate keys $sk_{(i'0^l 1),h}$ for all $0 \le l \le t - \tilde{k} - 1$, and $sk_{(i'0^{t-\tilde{k}},h)}$. Erase $sk_{i',h}$ and output the remaining keys as $SK_{(i+1),h}$.

ENCRYPT [1] $(i, (\mathrm{ID}_0, \ldots, \mathrm{ID}_h), M)$ (where $M \in \{0,1\}^n$):

1. Parse i as $i_1 \ldots i_t$. Select random $r \leftarrow \mathbb{Z}_q$.
2. Denote $P_{k,j} = H_1(i|_k \circ \mathrm{ID}_0 \ldots \mathrm{ID}_j)$ for all $1 \le k \le t$ and $0 \le j \le h$.

   Output $\langle i, (\mathrm{ID}_0, \ldots, \mathrm{ID}_h), C \rangle$, where

   $C = (rP, rP_{2,0}, \ldots, rP_{t,0}, rP_{1,1}, \ldots, rP_{t,1}, \ldots, rP_{1,h}, \ldots, rP_{t,h}, M \oplus H_2(\hat{e}(Q, H_1(i_1))^r))$.

DECRYPT[2] (i, C):

1. Parse i as $i_1 \ldots i_t$. Parses $SK_{i,h}$ associated with the ID-tuple as $(sk_{i,h}, \{sk_{(i|_k 1),h}\}_{i_k=0})$ and $sk_{i,h}$ as $(S_{i,h}, \mathcal{Q}_{i,h})$. Parse $\mathcal{Q}_{i,h}$ as: $\{Q_{k,j}\}$ for all $1 \le k \le t$ and $0 \le j \le h$, except $k = 1$ and $j = 0$.
2. Parse C as $(U_0, U_{2,0}, \ldots, U_{t,0}, U_{1,1}, \ldots, U_{t,1}, \ldots, U_{1,h}, U_{t,h}, V)$.
3. $M = V \oplus H_2(\frac{\hat{e}(U_0, S_{i,h})}{g})$, where g is: $\Pi_{k=1}^t \Pi_{j=1}^h \hat{e}(Q_{k,j}, U_{k,j}) \Pi_{k=2}^t \hat{e}(Q_{k,0}, U_{k,0})$.

We now verify that decryption of FHIBE is performed correctly. When encrypting, we have $\hat{e}(Q, H_1(i_1))^r = \hat{e}(P, H_1(i_1))^{r s_\epsilon}$. Recall $U_{kj} = rP_{kj} = rH_1(i_1 \ldots i_k \circ \mathrm{ID}_0 \ldots \mathrm{ID}_j)$.

$$
\begin{aligned}
\frac{\hat{e}(U_0, S_{i,h})}{\Pi_{k=1}^t \Pi_{j=1}^h \hat{e}(Q_{k,j}, U_{k,j}) \Pi_{k=2}^t \hat{e}(Q_{k,0}, U_{k,0})} &= \frac{\hat{e}(rP, s_\epsilon P_{1,0} + \sum_{k=1}^t \sum_{j=1}^h s_{k,j} P_{k,j} + \sum_{k=2}^t s_{k,0} P_{k,0})}{\Pi_{k=1}^t \Pi_{j=1}^h \hat{e}(s_{k,j} P, U_{k,j}) \Pi_{k=2}^t \hat{e}(s_{k,0} P, U_{k,0})} \\
&= \frac{\hat{e}(P, P_{1,0})^{r s_\epsilon} \Pi_{k=1}^t \Pi_{j=1}^h \hat{e}(rP, s_{k,j} P_{k,j}) \Pi_{k=2}^t \hat{e}(rP, s_{k,0} P_{k,0})}{\Pi_{k=1}^t \Pi_{j=1}^h \hat{e}(s_{k,j} P, rP_{k,j}) \Pi_{k=2}^t \hat{e}(s_{k,0} P, rP_{k,j})} \\
&= \hat{e}(P, P_{1,0})^{r s_\epsilon} \\
&= \hat{e}(P, H_1(i_1))^{r s_\epsilon}
\end{aligned}
$$

Because our FHIBE scheme builds on the HIBE scheme by Gentry and Silverberg [16], the construction of our FHIBE scheme is very similar to that of HIBE scheme.

[1] Using Fujisaki-Okamoto padding [14] and the help of random oracles H_3 and H_4, algorithm ENCRYPT can be converted into one with chosen ciphertext security, as in IBE [7] and HIBE [16] schemes.

[2] Using Fujisaki-Okamoto padding [14] and the help of random oracles H_3 and H_4, algorithm DECRYPT can be converted into one that correctly decrypts messages encrypted under chosen ciphertext security.

Theorem 1. *Suppose there is a nonadaptive adversary $\mathcal{A}$ that has advantage ϵ against the one-way secure FHIBE scheme for some fixed time t and ID-tuple that makes $q_{H_2} > 0$ hash queries to the hash function H_2 and a finite number of lower-level setup queries. If the hash functions H_1, H_2 are random oracles, then there is an algorithm $\mathcal{B}$ that solves the BDH in groups generated by $\mathcal{IG}$ with advantage $(\epsilon - \frac{1}{2^n})/q_{H_2}$ and running time $\mathcal{O}(time(\mathcal{A}))$.*

The proof of this theorem is in the Appendix.

4 Multiple Hierarchical Identity-based Encryption Scheme

In this section we generalize the FHIBE scheme into a collusion resistant multiple hierarchical identity-based encryption (MHIBE) scheme. The current HIBE [19, 16] schemes are suitable for encrypting a message under one identity. For example, Alice at the payroll office of Computer Science Department at some university has ID-tuple (University, CS, Payroll). Suppose Alice also works at the payroll office that is in charge of NSF funds at the same university. This identity of her may be expressed as another ID-tuple (University, NSF, Payroll). Bob knows both identities of Alice and wants to send an encrypted message so that only the person has both of the identities can decrypt it. Any person with only one of the two identities should be unable to read the message. Furthermore it cannot be decrypted by two persons, each with one of the identities, collude with each other. Clearly, the HIBE [16] scheme is unable to handle this situation.

An MHIBE scheme is useful when a message needs to be encrypted under multiple identities to ensure only the one with all the identities can decrypt it. We define an *identity-tuple* as the set of identities, represented as ID-tuples, that a user has. In this example, the identity tuple of Alice is {(University, CS, Payroll), (University, NSF, Payroll)}. In our MHIBE scheme, Alice may obtain her key from two entities. One is the entity whose identity-tuple is {(University, CS, Payroll), (University, NSF)}. The other has the identity-tuple of {(University, CS), (University, NSF, Payroll)}. Alice's key cannot be generated by the entity only having the identity of {(University, CS, Payroll)}, or {(University, NSF, Payroll)}. In the same way, Alice's parents obtain their keys from their parents. For example the entity with identity-tuple {(University, CS), (University, NSF)} receives its key from either {(University, CS), (University)} or {(University), (University, NSF)}.

A sender encrypts a message under the identity-tuple of Alice to ensure that only those who possess secret keys of both ID-tuples can decrypt the message. The sender does not have to know from which lower-level PKG Alice obtains her key and knowing the receiver's identity-tuple is sufficient to send a message. There is one subtlety in the MHIBE scheme. A message encrypted under all the identities of a user should not be decrypted by collusions. The collusion resistance requirement is necessary to prevent that a message sent to a user with two identities is decrypted by two other persons, each of whom happens to have just one of the identities. In this example, a message to Alice should not be able to decrypted by a person with identity-tuple {(University, CS, Payroll)} colluding with another person with identity-tuple {(University, NSF, Payroll)}.

Our FHIBE scheme naturally gives rise to an MHIBE scheme. In FHIBE scheme a message is encrypted under both an ID-tuple and the current time. The encryption of a message in FHIBE scheme can be thought of as the encryption under two ID-tuples, one of which being the current time. Therefore the identities in MHIBE scheme capture a broad sense of meaning. They represent the characteristics of the user at the time a message is encrypted. For example, an identity may be the user's geographic location, network address, or the current time, as in the case of FHIBE scheme.

The ability of our schemes to handle dynamic joins makes it easy to incorporate dynamic information as identities. Our MHIBE scheme generalized from FHIBE scheme requires the identities of a user have to share the same root PKG. The order of identities in an identity-tuple matters in the encryption and decryption, and is determined by how the user obtains his key. A message encrypted under $\{(\text{University}, \text{CS}), (\text{University}, \text{NSF})\}$ or $\{(\text{University}, \text{NSF}), (\text{University}, \text{CS})\}$ gives different ciphertext distribution, and requires different decryption keys. We note that in this MHIBE scheme, a user with the private key to multiple identities does not imply that he or she has the private key to any subset of the identities.

We omit the details of MHIBE scheme, as this is a direct generalization of FHIBE scheme. The MHIBE scheme built from our FHIBE scheme has total collusion resistance. We show the complexities of various parameters of our MHIBE scheme in Section 1.4.

References

[1] M. Abdalla, S. K. Miner, and C. Namprempre. Forward-secure threshold signature schemes. In *Topics in Cryptography – CT-RSA '01*, volume 2020 of *LNCS*. Springer-Verlag, 2001.

[2] R. Anderson. Two remarks on public-key cryptology. Invited lecture, CCCS '97. Available at `http://www.cl.cam.ac.uk/ftp/users/rja14/`.

[3] M. Bellare and S. Miner. Forward-secure digital signature scheme. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *LNCS*. Springer-Verlag, 1999.

[4] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In *Advances in Cryptology – Crypto '99*, volume 1666 of *LNCS*. Springer-Verlag, 1999.

[5] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[6] M. Bellare and B. Yee. Forward security in private-key cryptography. In *CT-RSA*, volume 2612 of *LNCS*. Springer-Verlag, 2003.

[7] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. Extended version of [8], `http://www.cs.stanford.edu/~dabo/papers/ibe.pdf`.

[8] D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology – CRYPTO '01*, volume 2139 of *LNCS*. Springer-Verlag, 2001.

[9] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. To appear in Contemporary Mathematics, American Mathematical Society, available at `http://eprint.iacr.org/2002/080/`.

[10] R. Canetti and S. Halevi. Simple forward-secure encryption. Available at `http://www.research.ibm.com/people/s/shaih/pubs/forward.html`.

[11] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. To appear in Advances in Cryptology – Eurocrypt '03. Available at `http://eprint.iacr.org/2003/083/`.

[12] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Advances in Cryptology – Crypto '89*, volume 435 of *LNCS*, pages 307–315. Springer-Verlag, 1989.

[13] W. Diffie, P. van Oorschot, and W. Wiener. Authentication and authenticated key exchanges. In *Designs, Codes and Cryptography*, volume 2, pages 107–125, 1992.

[14] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology – CRYPTO '99*, volume 1666 of *LNCS*. Springer-Verlag, 1999.

[15] C. Günther. An identity-based key exchange protocol. In *Advances in Cryptology - Eurocrypt '89*, volume 434 of *LNCS*, pages 29–37. Springer-Verlag, 1989.

[16] C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In *Advances in Cryptology – ASIACRYPT '02*, volume 2501 of *LNCS*. Springer-Verlag, 2002.

[17] C. Gentry and A. Silverberg. Hierarchical id-based cryptography. Cryptology ePrint Archive, Report 2002/056, 2002. Extended version of [16], `http://eprint.iacr.org/`.

[18] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In *Proceedings of the 4th Conference on Computer and Communications Security*, pages 100–110. ACM, 1997.

[19] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In *Advances in Cryptology – EUROCRYPT '02*, volume 2332 of *LNCS*. Springer-Verlag, 2002.

[20] S. Jarecki and A. Lysyanskaya. Concurrent and erasure-free models in adaptively secure threshold cryptography. In *Advances in Cryptology – EUROCRYPT '00*, LNCS, pages 221–242. Springer-Verlag, 2002.

[21] A. Joux. A one-round protocol for tripartite Diffie-Hellman. In *Proceedings of the ANTS-IV conference*, volume 1838, pages 385–394. Springer-Verlag, 2000.

[22] J. Katz. A forward-secure public-key encryption scheme. Cryptology ePrint Archive, Report 2002/060, 2002. `http://eprint.iacr.org/`.

[23] A. Lysyanskaya and C. Peikert. Adaptive security in the threshold setting: From cryptosystems to signature schemes. In *Advances in Cryptology – ASIACRYPT '01*, LNCS, pages 331–350. Springer-Verlag, 2001.

[24] T. Malkin, D. Micciancio, and S. K. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In *Advances in Cryptology – EUROCRYPT '02*, volume 2332 of *LNCS*. Springer-Verlag, 2002.

[25] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *10th Annual Symposium on Principles of Distributed Computing*, pages 51–59. ACM, 1991.

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

[27] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – Crypto '84*, volume 196 of *LNCS*. Springer-Verlag, 1984.

[28] N. P. Smart. Access control using pairing based cryptography. In *CT-RSA*, LNCS, pages 111–121. Springer-Verlag, 2003.

A Proof of Security

We give the proof of one-way security against nonadaptive chosen time and target adversary. **Phase 1** is omitted for the nonadaptive adversary in the one-way security definition.

Proofs for chosen-ciphertext security or one-way security against adaptive chosen-target adversary are not shown in this paper. They can be derived from the proof below using similar proof techniques as shown in IBE [7], HIBE [16] papers. The proof of one-way security against adaptive chosen-target adversary would be similar to the proof in HIBE [16] scheme, and make use of random coin flips at in H_1 queries. The number of coin flips needed for time node w and ID-tuple $(\mathrm{ID}_0, \ldots, \mathrm{ID}_h)$ corresponds to the level of the ID-tuple h and the length of the time node $|w|$. The proof of one-way security against adaptive chosen-target adversary may be converted to a proof of chosen-ciphertext security. Similar arguments to the ones in the proof of IBE [8] scheme would be used to show that the adversary $\mathcal{B}$ who uses an FHIBE adversary $\mathcal{A}$ to break BasicPub (shown below) does not abort with non-negligible probability.

A.1 BasicPub

To analyze the security of FHIBE scheme, we make use of a public-key encryption scheme called BasicPub [8], which was presented by Boneh and Franklin in the proof of their IBE scheme. Boneh and Franklin showed that BasicPub is one-way secure in the random oracle model under the BDH assumption [8]. To show that our FHIBE scheme is secure, we give a reduction from breaking FHIBE scheme to BasicPub. We use the techniques of Gentry and Silverberg in the proof of HIBE [16] scheme.

BasicPub is a public-key encryption scheme, specified by three algorithms: Generate Key, Encrypt and Decrypt.

Generate Key:

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

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

Encrypt: To encrypt $M \in \mathcal{M}$, do the following:

1. Choose a random $r \in \mathbb{Z}_q$.
2. Set the ciphertext to be: $C = (rP, M \oplus H_2(g^r))$ where $g = \hat{e}(Q, P_1) \in \mathbb{G}_2$.

Decrypt: Let $C = (U, V) \in \mathcal{C}$ be the ciphertext. To decrypt C, compute: $V \oplus H_2(\hat{e}(U, S_1)) = M$.

A.2 FHIBE scheme –Targeting a specific ID-tuple and time period

Lemma 1. *Let H_1 be a random oracle from $\{0,1\}^*$ to $\mathbb{G}_1$. Let $\mathcal{A}$ be a nonadaptive adversary against the one-way secure FHIBE scheme that makes a finite number of lower-level setup queries and has advantage ϵ against FHIBE for some fixed time t_0 and ID-tuple$_0$. Then there is an OWE adversary $\mathcal{B}$ that has advantage at least ϵ against BasicPub. Its running time is $\mathcal{O}(time(\mathcal{A}))$.*

Proof: Let us construct an OWE adversary $\mathcal{B}$ that uses $\mathcal{A}$ to gain advantage ϵ against BasicPub. The game between the challenger and the adversary $\mathcal{B}$ starts with the challenger first generating a random public key by running the key generation algorithm of BasicPub. The result is a public key $K_{pub} = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, P_1, H_2)$, with $Q = s_\epsilon P$, and a private key $S_1 = s_\epsilon P_1$. The challenger then picks a random plaintext $M \in \mathcal{M}$ and encrypts M using the encryption algorithm of BasicPub. It gives K_{pub} and the resulting ciphertext $C = (U, V)$ to algorithm $\mathcal{B}$. Algorithm $\mathcal{B}$ is supposed to output a guess for M.

Let the ID-tuple$_0$ and time period t_0 be the ID-tuple and the time period for which $\mathcal{A}$ has an advantage against FHIBE scheme. Then $\mathcal{A}$ and $\mathcal{B}$ agree to use the ID-tuple and the time period.

Before we go any further into details, let us first give an overview of what follows. $\mathcal{B}$ has essentially two goals to accomplish in order to succeed in this game. One is to be able to answer H_1 queries and lower-level setup queries made by $\mathcal{A}$, without knowing s_ϵ or $s_\epsilon P_1$. Recall that SK_{t_i,h_i} denotes the private key associated with time t_i and ID-tuple $(\mathrm{ID}_0 \ldots \mathrm{ID}_{h_i})$. It consists of a series of $(sk_{t_i,h_i}, \{sk_{w,h_i}\})$ values, each of which contains an S_{w,h_i} and $\mathcal{Q}_{w,h_i} = \{Q_{k,j}$ for all $1 \le k \le |w|$ and $0 \le j \le h_i\}$. A private key SK_{t_i,h_i} is *valid* if the following holds. Each $sk_{w,h}$ in SK_{t_i,h_i}, where w is any string, has to have the correct distribution, that is, $S_{w,h_i} = S_{w,(h_i-1)} + \sum_{k=1}^{|w|} s_{k,h_i} H_1(w|_k \circ \mathrm{ID}_0 \ldots \mathrm{ID}_{h_i}) = S_{(w|_{|w|-1}),h_i} + \sum_{j=0}^{h_i} s_{|w|,j} H_1(w \circ \mathrm{ID}_0 \ldots \mathrm{ID}_j) = \sum_{k=1}^{|w|} \sum_{j=0}^{h_i} s_{k,j} H_1(w|_k \circ \mathrm{ID}_0 \ldots \mathrm{ID}_j)$, and $Q_{k,j} = s_{k,j} P$ for some $s_{k,j} \in \mathbb{Z}_q$, for all $1 \le k \le |w|$ and $0 \le j \le h_i$. In the above expression, we let $S_{0,j}$ and $S_{k,-1}$ be the identity element in $\mathbb{G}_1$, for all j and k. Recall $w|_k$ denotes the first k-bit prefix of w, $w_1 \ldots w_k$.

The other goal of $\mathcal{B}$ in this proof is to convert his challenge ciphertext C into a FHIBE ciphertext C', so that it can be correctly decrypted by a private key of associated with the target t_0 and ID-tuple$_0$.

Setup: $\mathcal{B}$ gives $\mathcal{A}$ the FHIBE system parameters $params = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, H_1, H_2)$. H_1 is a random oracle controlled by $\mathcal{B}$. Let $N = 2^l$ be the total time periods. The target time period t_0 may be represented as $t_{0,1} \ldots t_{0,l}$.

H_1-queries: At any time, algorithm $\mathcal{A}$ can query the random oracle H_1. Essentially, this oracle will be used to store the Point-tuple$_i$ $= \{T_{i,k,j} = H_1(t_i|_k \circ \mathrm{ID}_{i,0} \ldots \mathrm{ID}_{i,j}) : 1 \le k \le l, 0 \le j \le h_i\}$ corresponding to Time-ID-tuple$_i$ $(t_i, (\mathrm{ID}_{i,0} \ldots \mathrm{ID}_{i,h_i}))$. In responding to these queries, algorithm $\mathcal{B}$ maintains a list H_1^{list} containing tuples of the form (Time-ID-tuple$_i$, Point-tuple$_i$, Scalar-tuple$_i$, Secret-tuple$_i$). This list is initially empty.

The time period t_0 together with the ID-tuple$_0$ creates a Time-ID-tuple, which are denoted as Time-ID-tuple$_0$. Algorithm $\mathcal{B}$ adds Time-ID-tuple$_0$ with to H_1^{list} as follows:

1. sets $s_{0,1,0} = 1$;
2. picks a random $s_{0,k,j} \in \mathbb{Z}_q$, for $1 \le k \le l$ and $0 \le j \le h_0$ except for $k = 1$ and $j = 0$;
3. picks a random $b_{0,k,j} \in \mathbb{Z}_q$, for $1 \le k \le l$ and $0 \le j \le h_0$;
4. sets $T_{0,1,0} = b_{0,1,0} P_1$ and $T_{0,k,j} = b_{0,k,j} P$ for $k \ge 1$ and $j \ge 0$ except for $k = 1$ and $j = 0$.

Algorithm $\mathcal{B}$ then puts the followings in H_1^{list}.

- $((t_{0,1}, \ldots, t_{0,l}), (\mathrm{ID}_{0,0}, \ldots, \mathrm{ID}_{0,h_0}))$
- $(T_{0,1,0}, \ldots, T_{0,1,h}, \ldots, T_{0,l,0}, \ldots, T_{0,l,h_0})$
- $(b_{0,1,0}, \ldots, b_{0,1,h_0}, \ldots, b_{0,l,0}, \ldots, b_{0,l,h_0})$
- $(s_{0,1,0}, \ldots, s_{0,1,h_0}, \ldots, s_{0,l,0}, \ldots, s_{0,l,h_0})$

Once that is done, $\mathcal{A}$ queries H_1 about Time-ID-tuple$_i$ $(t_i, (\mathrm{ID}_{i,0} \ldots \mathrm{ID}_{i,h_i}))$. Let us first briefly describe how $\mathcal{B}$ answers this query. $\mathcal{B}$ has to be ready in case $\mathcal{A}$ makes an lower-level setup query for this Time-ID-tuple$_i$. So an appropriate SK_{t_i,h_i} must be concocted. Recall that SK_{t_i,h_i} associated with time t_i and ID-tuple$_i$ $(\mathrm{ID}_{i,0}, \ldots, \mathrm{ID}_{i,h_i})$ consists of a series of $(sk_{t_i,h_i}, \{sk_{w,h_i}\})$ values, each of which contains an S_{w,h_i} and $\mathcal{Q}_{w,h_i} = \{Q_{k,j}$ for all $1 \leq k \leq |w|$ and $0 \leq j \leq h_i\}$.

For Time-ID-tuple$_i$ $(t_i, (\mathrm{ID}_{i,0} \ldots \mathrm{ID}_{i,h_i}))$, $\mathcal{B}$ answers H_1 query by computing the point-tuple values $T_{i,k,j} = H_1(t_i|_k \circ \mathrm{ID}_{i,0} \ldots \mathrm{ID}_{i,j}) : 1 \leq k \leq l, 0 \leq j \leq h_i\}$. $\mathcal{B}$ also chooses Scalar-tuple$_i$ and Secret-tuple$_i$ for Time-ID-tuple$_i$. Scalar-tuple$_i$, $\{b_{i,j,k}\}$ for all $1 \leq k \leq l, 0 \leq j \leq h_i$, is a set of values chosen from $\mathbb{Z}_q$. Secret-tuple$_i$, $\{s_{i,j,k}\}$ for all $1 \leq k \leq l, 0 \leq j \leq h_i$, is another set of values chosen from $\mathbb{Z}_q$. $s_{i,1,0} = 1$ for all i. Some of the values in Scalar-tuple$_i$ and Secret-tuple$_i$ for Time-ID-tuple$_i$ are inherited from some existing Time-ID-tuple in H_1^{list}, and others are freshly generated by $\mathcal{B}$.

At lower-level setup queries $\mathcal{B}$ uses these values to generate the key SK for some Time-ID-tuple. With in key SK, the value S_{t_i,h_i} is computed as the sum of the products of the values in the Secret-tuple and Point-tuple. Because of possible overlaps among the Time-ID-tuples, some of the values in some existing Point-tuple, Scalar-tuple and Secret-tuple associated with Time-ID-tuple$_i$ are used for Time-ID-tuple$_{i'}$, where $i' > i$.

In order for $\mathcal{B}$ to generate valid S values without knowing s_ϵ or $s_\epsilon P_1$, $\mathcal{B}$ does a trick at choosing one of the T values for Time-ID-tuple$_i$, so that this special T contains a term with negative sign that contains s_ϵ. These two terms with s_ϵ are conveniently canceled out with each other when computing S_{t_i,h_i}. $\mathcal{B}$ has to be careful at choosing this special T, which contains s_ϵ with negative sign, since only one such T is needed. In order to see if that trick is necessary for some Time-ID-tuple$_i$, $\mathcal{B}$ compares Time-ID-tuple$_i$ to both the target Time-ID-tuple$_0$ and the existing ones in H_1^{list} as follows.

For Time-ID-tuple $(t_i \circ \mathrm{ID}_{i,0} \ldots \mathrm{ID}_{i,h_i})$, let v be maximal such that $(t_{i,1}, \ldots, t_{i,x}) = (t_{g,1}, \ldots, t_{g,x})$ for some (Time-ID-tuple$_g$, Point-tuple$_g$, Scalar-tuple$_g$, Secret-tuple$_g$) already in H_1^{list}. Among such entries in H_1^{list}, choose the one with the maximum w such that $(\mathrm{ID}_{i,0}, \ldots, \mathrm{ID}_{i,w}) = (\mathrm{ID}_{g,0}, \ldots, \mathrm{ID}_{g,w})$.

Let $x \leq v$ be maximal such that $(t_{i,1}, \ldots, t_{i,x}) = (t_{0,1}, \ldots, t_{0,x})$, and y be maximal such that $(\mathrm{ID}_{i,0}, \ldots, \mathrm{ID}_{i,y}) = (\mathrm{ID}_{0,0}, \ldots, \mathrm{ID}_{0,y})$. Because in the FHIBE scheme ID_0 is the ID of the root PKG and it is the same for all ID-tuples, $y = 0$ means ID-tuple$_i$ and ID-tuple$_g$ share only ID_0, similar for w.

The purpose of these comparisons is to see if the trick of choosing the special T is needed. For the portion of Time-ID-tuple$_i$ that overlaps with existing Time-ID-tuple in H_1^{list}, $\mathcal{B}$ copies them from the existing Scalar, Secret and Point-tuples. For the part of the Time-ID-tuple that does not overlap with existing ones, $\mathcal{B}$ distinguishes four cases.

- If the time t_i in Time-ID-tuple$_i$ overlaps with t_g in Time-ID-tuple$_g$ more than with t_0 in Time-ID-tuple$_0$, there is no need to choose the special T value for Time-ID-tuple$_i$, since it is included in the T values inherited from Time-ID-tuple$_g$.

- If the Time-ID-tuple overlaps with Time-ID-tuple$_i$ most is Time-ID-tuple$_0$, $\mathcal{B}$ has to do the trick at choosing one special T value for Time-ID-tuple$_i$.
- If the time t_i in Time-ID-tuple$_i$, t_g in Time-ID-tuple$_g$, and t_0 in Time-ID-tuple$_0$ are the same, $\mathcal{B}$ compares their ID-tuples. The same way as comparing the time as above. If ID-tuple$_i$ overlaps more with ID-tuple$_g$ than with ID-tuple$_0$, $\mathcal{B}$ does not need to do the trick.
- Otherwise, $\mathcal{B}$ needs to do the trick and compute one special T value for Time-ID-tuple$_i$.

The details are as follows.

1. For $1 \leq k \leq v$ and $0 \leq j \leq w$, $\mathcal{B}$ sets $T_{i,k,j} = T_{g,k,j}$, $s_{i,k,j} = s_{g,k,j}$, and $b_{i,k,j} = b_{g,k,j}$.
2. If $0 \leq x < v \leq l$, then for $v < k \leq l$ or $w < j \leq h_i$, algorithm $\mathcal{B}$:
   (a) picks a random $s_{i,k,j} \in \mathbb{Z}_q$;
   (b) picks a random $b_{i,k,j} \in \mathbb{Z}_q$;
   (c) sets $T_{i,k,j} = b_{i,k,j}P$.
3. If $0 \leq x = v < l$, then for $x < k \leq l$ or $w < j \leq h_i$, algorithm $\mathcal{B}$:
   (a) picks a random $s_{i,k,j} \in \mathbb{Z}_q$;
   (b) picks a random $b_{i,k,j} \in \mathbb{Z}_q$;
   (c) sets $T_{i,(x+1),0} = b_{i,(x+1),0}P - s^{-1}_{i,(x+1),0}b_{i,1,0}P_1$;
   (d) for $1 \leq k \leq l$ and $w+1 \leq j \leq h_i$, sets $T_{i,k,j} = b_{i,k,j}P$;
   (e) for $x < k \leq l$ and $0 \leq j \leq h_i$ except $k = x+1$ and $j = 0$, sets $T_{i,k,j} = b_{i,k,j}P$.
4. If $x = v = l$ and $0 \leq y < w < h_i$, then for $1 < k \leq l$ and $w < j \leq h_i$, algorithm $\mathcal{B}$:
   (a) picks a random $s_{i,k,j} \in \mathbb{Z}_q$;
   (b) picks a random $b_{i,k,j} \in \mathbb{Z}_q$;
   (c) sets $T_{i,k,j} = b_{i,k,j}P$.
5. If $x = v = l$ and $0 \leq y = w < h_i$, then for $1 < k \leq l$ and $w < j \leq h_i$, algorithm $\mathcal{B}$:
   (a) picks a random $s_{i,k,j} \in \mathbb{Z}_q$;
   (b) picks a random $b_{i,k,j} \in \mathbb{Z}_q$;
   (c) sets $T_{i,1,(y+1)} = b_{i,1,(y+1)}P - s^{-1}_{i,1,(y+1)}b_{i,1,0}P_1$;
   (d) for $1 \leq k \leq l$ and $y+1 \leq j \leq h_i$ except $k = 1$ and $j = y+1$, sets $T_{i,k,j} = b_{i,k,j}P$.

Algorithm $\mathcal{B}$ then puts the following in H_1^{list} and returns $(T_{i,1,0}, \ldots, T_{i,l,h_i})$ to $\mathcal{A}$.

- $((t_{i,1}, \ldots, t_{i,l}), (\mathrm{ID}_{i,0}, \ldots, \mathrm{ID}_{i,h_i}))$
- $(T_{i,1,0}, \ldots, T_{i,l,h_i})$
- $(b_{i,1,0}, \ldots, b_{i,l,h_i})$
- $(s_{i,1,0}, \ldots, s_{i,l,h_i}))$

Note that $T_{i,k,j}$ is always chosen uniformly in $\mathbb{G}_1$ and is independent of $\mathcal{A}$'s view as required.

Using above methods, $\mathcal{B}$ also stores the corresponding tuples that are necessary to compute the node keys at lower-level setup queries. Recall for time t_i and ID-tuple$_i$ $(\mathrm{ID}_0, \ldots, \mathrm{ID}_{h_i})$, node keys are denoted as sk_{w,h_i}, where w are all the labels of the right sibling, if one exists, of each ancestor of the node labelled t_i in the complete binary tree. They can also be written as $\{sk_{(t_i|_k 1),h_i}\}_{t_{i,k}=0}$. For each time node w, $\mathcal{B}$ compares $(w_1 \ldots w_{|w|})$ with existing tuples in H_1^{list}, the same way as comparing time t_i. Node keys together with the decryption key sk_{t_i,h_i} are what $\mathcal{B}$ needs to return in an lower-level setup query. We omit the details of how $\mathcal{B}$ prepares for node keys in this proof, as it is essentially the same as parparing the decryption key shown above.

Lower-Level Setup Queries: At any time, $\mathcal{A}$ may make an lower-level setup query on any time period t_i and ID-tuple$_i$, given $t_i > t_0$, or ID-tuple$_i \neq$ ID-tuple$_0$ and its ancestor. $\mathcal{B}$ responds to this query as follows:

1. Run the above algorithm for responding to H_1-queries to obtain the appropriate tuple (Time-ID-tuple$_i$, Point-tuple$_i$, Scalar-tuple$_i$, Secret-tuple$_i$) in H_1^{list}.
2. Define $S_{i,t_i,h_i} = \sum_{k=1}^{l} \sum_{j=0}^{h_i} s_{i,k,j} s_\epsilon T_{i,k,j}$ where

   $s_{i,1,0} := 1$ for all i. $\mathcal{B}$ also gives $\mathcal{A}$ the corresponding

   $\{Q_{i,k,j} = s_{i,k,j} Q : 1 \leq k \leq l, 0 \leq j \leq h_i\}$.

   Let $sk_{i,t_i,h_i} = (S_{i,l,h_i}, \mathcal{Q}_{i,t_i,h_i})$, where $\mathcal{Q}_{i,t_i,h_i} = \{Q_{i,k,j} = s_{i,k,j} Q : 1 \leq k \leq l, 0 \leq j \leq h_i\}$. sk_{i,t_i,h_i} is what ID-tuple$_i$ needs to decrypt messages at t_i.
3. Using similar method for computing sk_{i,t_i,h_i}, $\mathcal{B}$ also computes the node keys $\{sk_{i,(t_i|_k 1),h_i}\}_{t_i|_k=0}$ corresponding to time t_i and ID-tuple$_i$. It can be verified that those node keys have the correct distribution. Using these node keys $\mathcal{B}$ can derive keys SK that are associated with any time $t > t_i$ or descendants of ID-tuple$_i$ with the correct distribution.

We leave to the readers the verification that shows the private key, in particular S_{t_i,h_i} for Time-ID-tuple$_i$ is computable by $\mathcal{B}$ and has the correct distribution. Note that $\mathcal{B}$ does not know s_ϵ nor $s_\epsilon P_1$.

Challenge: At any time, $\mathcal{A}$ may request a challenge ciphertext from $\mathcal{B}$ on the ID-tuple$_0$ and time period t_0. Let $C = (U, V)$ be the challenge ciphertext given to algorithm $\mathcal{B}$. $\mathcal{B}$ sets the FHIBE ciphertext C' to be

$$
\begin{gathered}
(b_{0,1,0}^{-1} U, b_{0,1,0}^{-1} b_{0,1,1} U, \ldots, b_{0,1,0}^{-1} b_{0,1,h_0} U, \\
b_{0,1,0}^{-1} b_{0,2,0} U, b_{0,1,0}^{-1} b_{0,2,1} U, \ldots, b_{0,1,0}^{-1} b_{0,2,h_0} U, \\
\ldots \\
b_{0,1,0}^{-1} b_{0,l,0} U, b_{0,1,0}^{-1} b_{0,l,1} U, \ldots, b_{0,1,0}^{-1} b_{0,l,h_0} U, V)
\end{gathered}
$$

$\mathcal{B}$ responds to $\mathcal{A}$ with the challenge C'. Note that C' is an FHIBE encryption of M under ID-tuple$_0$ and t_0 as required. To verify this, first observe that the decryption key corresponding to ID-tuple$_0$ and t_0 is $S' = s_\epsilon T_{0,1,0} + \sum_{j=1}^{h_0} \sum_{k=1}^{l} s'_{k,j} T_{0,k,j} + \sum_{k=2}^{l} s'_{k,0} T_{0,k,0}$ along with the additional information $\{s'_{k,j} P : 1 \leq k \leq l \text{ and } 0 \leq j \leq h_0\}$ (except $s'_{1,0}$) for some set $\{s'_{k,j} : 1 \leq k \leq l \text{ and } 0 \leq j \leq h_0\}$ (except $s'_{1,0}$).

Second, observe that :

$$\begin{aligned}
\frac{\hat{e}(b_{0,1,0}^{-1}U, S)}{\Pi_{k=1}^{l}\Pi_{j=1}^{h_0}\hat{e}(b_{0,1,0}^{-1}b_{0,k,j}U, s'_{k,j}P)\Pi_{k=2}^{l}\hat{e}(b_{0,1,0}^{-1}b_{0,k,0}U, s'_{k,0}P)} &= \hat{e}(b_{0,1,0}^{-1}U, s_\epsilon T_{0,1,0}) \\
&= \hat{e}(b_{0,1,0}^{-1}U, s_\epsilon T_{0,1,0}) \\
&= \hat{e}(b_{0,1,0}^{-1}U, s_\epsilon b_{0,1,0}P_1) \\
&= \hat{e}(U, s_\epsilon P_1)
\end{aligned}$$

The correct decryption of C' is M.

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

Claim: $\mathcal{A}$'s view is identical to its view in the real attack. Furthermore, $\Pr[M = M'] \geq \epsilon$. The probability is over the random bits used by $\mathcal{A}$, $\mathcal{B}$ and the challenger.

Proof of Claim: All responses to H_1-queries are as in the real attack since each response is uniformly and independently distributed in $\mathbb{G}_1$. All responses to lower-level setup queries are valid. Finally, the challenge ciphertext C' given to $\mathcal{A}$ is the FHIBE encryption of the random plaintext M under the ID-tuple$_0$ and time period t_0 chosen by $\mathcal{A}$. Therefore, by the definition of the algorithm $\mathcal{A}$, it will output $M' = M$ with probability at least ϵ.

A.3 Security Proof for FHIBE scheme

To complete the proof for FHIBE scheme, we need the following result, which is Lemma 4.3 of [7], says that solving BDH reduces to breaking BasicPub.

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

Theorem 1 for FHIBE scheme follows by combining Lemma 1 and Lemma 2.