Forward-Secure Hierarchical IBE with Applications to Broadcast Encryption [1]

Danfeng (Daphne) YAO [a,2] Nelly FAZIO [b] Yevgeniy DODIS [c]
Anna LYSYANSKAYA [d]

[a] *Department of Computer Science, Rutgers University, New Brunswick, NJ*
[b] *IBM Almaden Research Center, San Jose, CA*
[c] *Department of Computer Science, New York University, NY*
[d] *Department of Computer Science, Brown University, Providence, RI*

Abstract. A forward-secure encryption scheme protects secret keys from exposure by evolving the keys with time. Forward security has several unique requirements in hierarchical identity-based encryption (HIBE) scheme: (1) users join dynamically; (2) encryption is joining-time-oblivious; (3) users evolve secret keys autonomously.

We define and construct a scalable pairing-based forward-secure HIBE (fs-HIBE) scheme satisfying all of the above requirements. We also show how our fs-HIBE scheme can be used to realize a forward-secure public-key broadcast encryption scheme, which protects the secrecy of prior transmissions in the broadcast encryption setting. We further generalize fs-HIBE into a collusion-resistant multiple hierarchical ID-based encryption scheme, which can be used for secure communications with entities having multiple roles in role-based access control. The security of our schemes is based on the bilinear Diffie-Hellman assumption in the random oracle model.

Keywords. Forward security, ID-based encryption, broadcast encryption

1. Introduction

The idea of an identity-based encryption (IBE) scheme is that an arbitrary string can serve as a public key. The main advantage of this approach is to largely reduce the need for public key certificates and certificate authorities, because a public key is associated with identity information such as a user's email address. A first scheme for identity-based encryption (BF-IBE) was based on the bilinear Diffie-Hellman assumption in the random oracle model by Boneh and Franklin [10]. In IBE schemes private key generator (PKG) is responsible for generating private keys for all users, and therefore is a performance bottleneck for organizations with large number of users. Hierarchical identity-based encryption (HIBE) schemes [7,20,24] were proposed to alleviate the workload of a root PKG by delegating private key generation and identity authentication to lower-level

[1] The preliminary version of this chapter has been published in the Proceedings of the ACM Conference on Computer and Communications Security (CCS '04) [38].

[2] Corresponding Author: Department of Computer Science, Rutgers University, New Brunswick, RI 08854; E-mail: danfeng@cs.rutgers.edu.

PKGs. In a HIBE scheme, a root PKG needs only to generate private keys for domain-level PKGs, who in turn generate private keys for users in their domains in the next level. The organization of PKGs and users forms a hierarchy that is rooted by the root PKG. To encrypt a message, Alice needs to obtain the public parameters of Bob's root PKG, and the ID for Bob and for those domain-level PKGs that are on the path from the root to Bob; there are no lower-level parameters. Gentry and Silverberg [20] extended BF-IBE scheme and presented a fully scalable hierarchical identity-based encryption (GS-HIBE) scheme. Later, a HIBE construction with a weaker notion of security was given by Boneh and Boyen [7]. Most recently, new IBE and HIBE constructions that can be proved to have the full security without the random oracle model [8,35] were given.

Due to the inherent key-escrow property, the standard notion of HIBE security crucially depends on secret keys remaining secret. Key exposure is a realistic threat over the lifetime of such a scheme. To mitigate the damage caused by the exposure of secret key information in HIBE, one way is to construct a forward-secure hierarchical identity-based encryption (fs-HIBE) scheme that allows each user in the hierarchy to refresh his or her private keys periodically while keeping the public key the same. A forward-secure public-key encryption scheme has recently been presented by Canetti, Halevi and Katz [12]. But surprisingly, a practical fs-HIBE scheme has several unique requirements that cannot be achieved by trivial combinations of the existing fs-PKE schemes [12,25] and HIBE scheme [7,20].

Apart from being interesting on its own, fs-HIBE is a useful tool that lends itself to several applications. One such application is the implementation of forward secrecy for public-key broadcast encryption. While forward secrecy is an important requirement in any context, it is especially needed for broadcast encryption [6,16,18,27,36]. This is because by design an adversary can freely listen to any broadcast and store it. Then, should the adversary ever succeed in recovering *any* user's secret key, she will manage to decrypt all past broadcasts that such user was authorized to receive *unless* we have forward secrecy.

In our preliminary version [38], we posed an interesting open question that whether a general fs-HIBE scheme with linear or even sub-linear complexity can be realized. Shortly afterward, Boneh, Boyen, and Goh were able to construct an efficient HIBE system with constant-size ciphertexts [9] under a different security assumption. Their HIBE scheme can also be extended to achieve forward-security with constant ciphertexts. The security of Boneh, Boyen, and Goh's system is based on a weaker version of Diffie-Hellman Inversion (BDHI) assumption. The BDHI assumption was previously used to construct a selective-ID secure IBE without random oracles [7]. In comparison, our system is only based on the Bilinear Diffie-Hellman (BDH) assumption. The 1-BDHI assumption is equivalent to the standard BDH assumption. It is not known if the h-BDHI assumption, for $h > 1$, is equivalent to BDH [7].

Below, we discuss the notion of forward security for HIBE in more detail, and then explain why it cannot be trivially achieved by existing techniques such as a combination of fs-PKE [12] and HIBE [7,20] schemes.

1.1. Forward Security

The central idea of forward secrecy is that the compromise of long-term keys does not compromise past session keys and therefore past communications. This notion was first proposed by Günther [19] and later by Diffie *et al.* [13] in key exchange protocols. The notion of non-interactive forward security was proposed by Anderson [2] in 1997 and later formalized by Bellare and Miner [3], who also gave a forward-secure signature scheme followed by a line of improvement [1,28]. 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 Yee [5] provided a comprehensive treatment of forward security in the context of private key based cryptographic primitives.

The first forward-secure public-key encryption (fs-PKE) scheme was given by Canetti, Halevi, and Katz [12] based on the Gentry-Silverberg HIBE [20] scheme. The fs-PKE scheme constructs a binary tree, in which a tree node corresponds to a time period and has a secret key. Children of a node w are labeled $w0$ and $w1$, respectively. Given the secrets corresponding to a prefix of a node representing time t, one can compute the secrets of time t. In order to make future keys computable from the current key, the secrets associated with a prefix of a future time are stored in the current key. After the key for the next time period is generated, the current decryption key is erased. The state-of-the-art fs-PKE scheme [12] is based on the decisional bilinear Diffie-Hellman assumption [10] in the standard model. Canetti, Halevi and Katz also gave a more efficient scheme in the random oracle model [12].

1.2. Requirements of a fs-HIBE Scheme

Intuitively, forward security in a HIBE scheme implies that compromise of the current secret key of a user only leads to the compromise of the user and his descendants' subsequent communications. We will give a formal definition of security in Section 2.2. Our design of a forward-secure HIBE scheme also takes system properties such as scalability and efficiency into consideration. This is essential in the management of large scale distributed systems. Below, we define the requirements for a scalable forward-secure HIBE scheme.

- New users should be able to join the hierarchy and receive secret keys from their parent nodes *at any time*.
- Encryption is *joining-time-oblivious*, which means that the encryption does not require knowledge of when a user or any of his ancestors joined the hierarchy. The sender can encrypt the message as long as he knows the current time and the ID-tuple of the receiver, along with the public parameters of the system.
- The scheme should be forward-secure.
- Refreshing secret keys can be carried out *autonomously*, that is, users can refresh their secret keys on their own to avoid any communication overhead with any PKG.

Surprisingly, the design of a fs-HIBE scheme that fulfils the above system requirements turns out to be non-trivial, despite the fact that both HIBE [20] scheme and fs-PKE [12] scheme are known. Intuitive combinations of the two schemes fail to achieve all the desired system features. Next, we explain why this is the case.

1.3. Some Forward-Secure HIBE Attempts

In this section, we make three simple forward-secure HIBE constructions based on HIBE scheme [20] and fs-PKE scheme [12], and explain why these naive schemes do not satisfy the requirements of a practical fs-HIBE scheme.

1.3.1. Scheme I

Consider a scheme based on the HIBE [20] scheme. The user with a given ID tuple $(ID_1, \ldots, ID_h)$ maintains two sub-hierarchies (subtrees): the time subtree that evolves over time for forward security (as in fs-PKE [12]), and the ID subtree to which other nodes are added as children join the hierarchy. To encrypt a message for this user at time t, use the HIBE with identity $(ID_1, \ldots, ID_h, t)$. The user can decrypt this message using HIBE decryption, using the fact that he knows the key from the time subtree. The user's children are added to the hierarchy into the ID subtree.

However, Scheme I has the following issue. Suppose a user never erases the secret key corresponding to the root of his ID subtree. Then should this key ever be exposed, the forward secrecy of his children is compromised. On the other hand, if this secret key is ever erased, then no nodes can be added as children of $(ID_1, \ldots, ID_h)$ in the hierarchy, and so this scheme will not support dynamic joins.

The lesson we learn from this failed scheme is that all keys must be evolved together.

1.3.2. Scheme II

Let us try to repair Scheme I by making sure that the key from which children's keys are derived is also evolving over time. In Scheme II, the public key of a user consists of alternating ID-tuples and time strings, which is referred to as an *ID-time-tuple*. The private key of a user serves three purposes: decryption, generating private keys for new children, and deriving future private keys of the user. The public key of a newly joined child is the parent's ID-time-tuple appended with the child's ID. That key is in turn used for generating keys for lower-level nodes further down the hierarchy. For example, if Alice joins Bob, the root, at time (*January, Week 1*) and Eve joins Alice at time (*January, Week 2*), Eve's public key is (Bob, *January, Week 1*, Alice, *January, Week 2,* Eve). Encrypting a message to Eve requires the sender to know when Eve and all her ancestors joined the system. Therefore Scheme II is not joining-time-oblivious.

The lesson we learn from the failed Scheme II is that the keys must evolve in a way that is transparent to the encryption algorithm.

1.3.3. Scheme III

In our final unsuccessful attempt, Scheme III, a user adds a child to the hierarchy by giving him or her secret keys that depend both on the current time and on the child's position in the hierarchy. This is achieved by requiring that messages may only be decrypted by those who know two keys: one corresponding to the current time and the other corresponding to their positions in the hierarchy. Each user autonomously evolves his time key, and gives his newly joined children his time key in addition to their ID keys.

It is easy to see that this scheme is *n*ot forward-secure. An adversary who joins the hierarchy at the beginning of time can corrupt a user at any future time and obtain his or her ID key. Moreover, this adversary can derive any past time key (because he joined at the beginning of time). Thus, this adversary may decrypt any past message addressed to the exposed user.

For the same reason, the multiple hierarchical identity-based encryption (MHIBE) scheme generalized from Scheme III is not collusion-resistant, where the ciphertext for a user with multiple identities can be decrypted if some other individuals collude. MHIBE scheme is useful for secure communications with entities having multiple identities, and is described in Section 1.4.3 and 5.

1.3.4. Comparisons

All the above trivial approaches fail. Constructing a forward-secure hierarchical ID-based encryption scheme that is both secure and scalable is not so straightforward. Our implementation, which is described in next Section, is still based on GS-HIBE [20] scheme and fs-PKE [12] scheme. Yet, it overcomes the problems existing in naive combinations of the two schemes, and satisfies the requirements of supporting dynamic joins, joining-time-obliviousness, forward security, and autonomous key updates.

1.4. Overview

We describe several cryptographic constructions. First, we present a scalable and joining-time-oblivious forward-secure hierarchical identity-based encryption scheme that allows keys to be updated autonomously. Second, we show how our fs-HIBE scheme can be used to obtain a forward-secure public-key broadcast encryption (fs-BE) scheme. Third, we generalize our fs-HIBE scheme and discuss its application in secure communications with entities having multiple roles in role-based access control (RBAC) [32].

1.4.1. Forward-Secure HIBE Scheme

Our fs-HIBE protocol is based on the HIBE scheme by Gentry and Silverberg [20] and forward-secure public-key encryption (fs-PKE) [12] scheme due to Canetti, Halevi and Katz. It satisfies the requirements of dynamic joins, joining-time-obliviousness, forward security, and autonomous key updates.

A HIBE scheme involves only one hierarchy, whereas a fs-HIBE scheme has two hierarchies: ID and time. Each (ID-tuple, time) pair can be thought of as a point on the two-dimensional grid as follows. On the x-axis, we start with the identity of the root Public Key Generator in the ID hierarchy (e.g. Hospital), then in position (1,0) we have the identity of the first-level PKG (e.g. ER). In position (2,0) there is the identity of the second level PKG (e.g. Doctor), and in position (3,0) there may be another PKG or an individual user (e.g. Bob). Thus the x-axis represents an ID-tuple, for example (Hospital, ER, Doctor, Bob). Similarly, the y-axis represents the time. Divide a duration of time into multiple time periods and arrange them as leaf nodes of a tree. Internal nodes of the tree represent the time spans associated with their child nodes. Then, the origin of the grid corresponds to the root of the time hierarchy (e.g. 2005). In position (0, 1) we have the first level of the time hierarchy (e.g. January), and in position (0, 2) there is the next level of time hierarchy (e.g. Week 1). Thus a time period can be expressed as a tuple on the y-axis, for example (2005, January, Week 1). Figure 1 gives a schematic drawing of the correspondence between the tuples and keys in fs-HIBE.

In a fs-HIBE scheme, the secret key of an (ID-tuple, time) pair is associated with some path on the grid. For each grid point on that path, there is a corresponding element

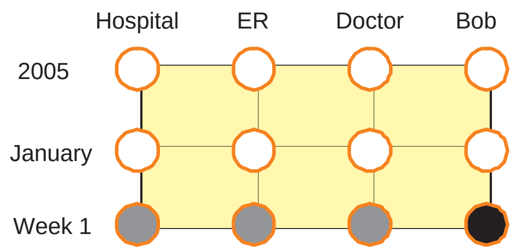

Figure 1. A schematic drawing of keys for ID-tuple (Hospital, ER, Doctor, Bob) at time period (2005, January, Week 1) in a forward-secure HIBE scheme. The ID-tuple (Hospital, ER, Doctor, Bob) of Bob is on x-axis. The tuple representing time period (2005, January, Week 1) is on y-axis. The origin represents the root identity (Hospital) and the highest-level time period (2005). The black node represents Bob's key at Week 1. The gray nodes correspond to keys of Bob's ancestors at Week 1. Each white node represents an intermediate key. Secret keys at both the grey and white nodes can be used to compute private keys for Bob.

in this secret key. Such a path (secret key) is *not* joining-time-oblivious: it depends on when the user, as well as the nodes higher up, join the system. However, when encrypting, the sender does not have to know the path. What is non-trivial here is that, the path (secret key) and ciphertext of our fs-HIBE scheme are designed in such a way that we do not need to come up with a separate ciphertext for each possible path in order to achieve joining-time-obliviousness.

Our fs-HIBE scheme has collusion resistance and chosen ciphertext security in the random oracle model [4] assuming the difficulty of the bilinear Diffie-Hellman problem [10,12,20], provided that the depths of the ID hierarchy and time hierarchy are bounded by constants. Our fs-HIBE scheme is provable secure under full-identity chosen-ciphertext model (ind-id-cca). The complexities of various parameters in our fs-HIBE scheme are summarized in Table 1 and are discussed in Section 6.

1.4.2. Forward-Secure Broadcast Encryption Scheme

We show how our fs-HIBE scheme can be used to construct a scalable forward-secure public-key broadcast encryption (fs-BE) scheme, which protects the secrecy of prior transmissions. A broadcast encryption (BE) [14,15,21,22,26,29,30,34] scheme allows content providers to securely distribute digital contents to a dynamically changing user population. Each active user is issued a distinct secret key when he joins the system, by a trusted center. In comparison with the symmetric-key setting, a public-key BE scheme of [14] has a single public key associated with the system, which allows the distribution of the broadcast workload to untrusted third parties.

In a scalable forward-secure public-key broadcast encryption (fs-BE) scheme, users should be able to update their secret keys autonomously, and the trusted center should allow users to dynamically join the broadcast system at any time while achieving forward security. In addition, each content provider does not need to know when each user joins the system in order to broadcast the encrypted contents. The encryption algorithm of a fs-BE scheme should only depend on the current time and the set of authorized users, and thus be joining-time-oblivious. Applying our fs-HIBE to the public-key BE scheme [14] yields such a fs-BE scheme.

1.4.3. Multiple Hierarchical ID-Based Encryption

We further generalize our forward-secure hierarchical ID-based encryption scheme into a collusion-resistant multiple hierarchical identity-based encryption (MHIBE) scheme, and describe its application in secure communications with individuals who have multiple roles in role-based access control (RBAC) [32]. In large-scale organizations, a user may own multiple identities, each of which is represented by an ID-tuple. In MHIBE, a message can be encrypted under multiple ID-tuples (identities) and can be decrypted only by those who have *all* the required identities. The collusion-resistant property cannot be achieved using separate HIBE schemes. We note that the fs-HIBE scheme is a special case of our MHIBE scheme, in that in fs-HIBE scheme, time can be viewed as another identity of a user. Therefore the identities in MHIBE scheme capture a broad sense of meaning.

2. Forward-secure HIBE (fs-HIBE)

This section defines the notion of forward secrecy for HIBE scheme and the related security. In a fs-HIBE scheme, secret keys associated with an ID-tuple are evolved with time. At any time period i an entity joins 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 joined entity receives this secret information, at the end of each period it updates its secret key and erases the old key. During time period i, a message is encrypted under an ID-tuple and the time i. Decryption requires the secret key of the ID-tuple at time i.

Time Period: As usual in forward-secure public-key encryption [12] scheme, we assume for simplicity that the total number of time periods N is a power of 2; that is $N = 2^l$. **ID-tuple:** An entity has a position in the hierarchy, defined by its tuple of IDs: $(\mathrm{ID}_1, \ldots, \mathrm{ID}_h)$. The entity's ancestors in the hierarchy are the users / PKGs whose ID-tuples are $\{(\mathrm{ID}_1, \ldots, \mathrm{ID}_i): 1 \leq i < h\}$. ID_1 is the ID for the root PKG.

2.1. fs-HIBE: Syntax

Forward-secure Hierarchical ID-Based Encryption Scheme (fs-HIBE): a fs-HIBE scheme is specified by five algorithms: Setup, KeyDer, Upd, Enc, and Dec.

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 $SK_{0,1}$. 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.

KeyDer: This algorithm is run by the parent of a newly joined child at time i to compute the child's private key. 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 $h-1$ computes the entity's key $SK_{i,h}$ associated with time period i. The inputs are the parent's private key $SK_{i,h-1}$, time i, and the ID-tuple of the child.

Upd: During the time period i, an entity (PKG or individual) with ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_h)$ uses $SK_{i,h}$ to compute his key $SK_{(i+1),h}$ for the next time period $i+1$, and erases $SK_{i,h}$.

Enc: 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 $C \in \mathcal{C}$.
Dec: During the time period i, a user with the ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_h)$ inputs $params$, $C \in \mathcal{C}$, and its secret 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 $SK_{i,h}$ is the secret key generated by algorithm KeyDer for ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_h)$ and time period i, then: $\forall M \in \mathcal{M}$, decryption of the ciphertext C with $params$ and the key $SK_{i,h}$ yields the message M, where C is the result of the encryption of M under time i and $(\mathrm{ID}_1, \ldots, \mathrm{ID}_h)$.

2.2. fs-HIBE: Security

We allow an attacker to make *key derivation queries*. Also, we allow the adversary to choose the time period and the identity on which it wishes to be challenged. Notice that an adversary may *choose* the time period and the identity of its targets adaptively or non-adaptively. An adversary that chooses its targets adaptively first makes key derivation queries and decryption queries, and then chooses 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 queries he makes. Security against an adaptive-chosen-target adversary, which is captured below, is the stronger notion of security than the non-adaptive one. It is also stronger than the selective-node security defined in the fs-PKE scheme by Canetti *et al.* [12].
Full-identity chosen-ciphertext security (ind-id-cca**)**: We say a fs-HIBE scheme is semantically secure against adaptive chosen ciphertext, time period, and identity 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 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[3]:

1. Key derivation query $(t_i, \text{ID-tuple}_i)$: the challenger runs the KeyDer algorithm to generate the private key $SK_{(t_i,\ \text{ID-tuple}_i)}$ corresponding to $(t_i, \text{ID-tuple}_i)$, and sends $SK_{(t_i, \text{ID-tuple}_i)}$ to the adversary.
2. Decryption query $(t_i, \text{ID-tuple}_i, C_i)$: the challenger runs KeyDer algorithm to generate the private key $SK_{(t_i, \text{ID-tuple}_i)}$ corresponding to the pair $(t_i, \text{ID-tuple}_i)$, runs the Dec algorithm to decrypt C_i using $SK_{(t_i, \text{ID-tuple}_i)}$, and sends the resulting plaintext to the adversary.

These queries may be asked adaptively. Also, the queried ID-tuple_i may correspond to a position at any level in the ID hierarchy, and the adversary is allowed to query for a future time and then for a past time.
Challenge: Once the adversary decides that **Phase 1** is over, it outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$, a time period t^* and an ID-tuple* on which it wishes to be

[3] In the random oracle model, the adversary may also issue public key queries. Public key query $(t_i, \text{ID-tuple}_i)$: challenger runs a hash algorithm on $(t_i, \text{ID-tuple}_i)$ to obtain the public key $H(t_i \circ \text{ID-tuple}_i)$ corresponding to $(t_i, \text{ID-tuple}_i)$, where H is a random oracle.

challenged. The constraint is that no key derivation query has been issued for ID-tuple* or any of its ancestors for any time $t \leq t^*$.

The challenger picks a random bit $b \in \{0, 1\}$, and sets $C^* = \mathsf{Enc}(params, t^*, \text{ID-tuple}^*, M_b)$. 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[4]:

1. Key derivation query $(t_i, \text{ID-tuple}_i)$, where the time period t_i and ID-tuple$_i$ are under the same restriction as in **Challenge**: the challenger responds as in **Phase 1**.
2. Decryption query $(t_i, \text{ID-tuple}_i, C_i) \neq (t^*, \text{ID-tuple}^*, C^*)$: the challenger responds as in **Phase 1**.

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

3. A Forward-secure HIBE Scheme

Here, we present a forward-secure hierarchical identity-based encryption scheme. Following the presentation standard in the IBE literature [10,20], we first present a fs-HIBE with *one-way security*. One-way security is the weakest notion of security. It means that it is hard to recover a plaintext with a passive attack. A standard technique, due to Fujisaki and Okamoto [17], converts one-way security to chosen-ciphertext security in the random oracle model. The definition of one-way security and the Fujisaki-Okamoto conversion of the one-way secure fs-HIBE can be found in [37].

Our scheme, which is based on the HIBE scheme of Gentry and Silverberg [20] and the fs-PKE scheme of Canetti, Halevi and Katz [12,25], overcomes the scalability and security problems that exist when naively combining the two schemes as described in Section 1.3. Next, we first give the number theoretic assumptions needed in our scheme, and then describe the algorithms in our construction.

3.1. Assumptions

The security of our fs-HIBE scheme is based on the difficulty of the bilinear Diffie-Hellman (BDH) problem [10]. 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 [10], we call $\hat{e}$ an admissible pairing if $\hat{e}\colon \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ 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 $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$.
3. Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}_1$.

[4] In the random oracle model, the adversary may also issue public key query. Public key query $(t_i, \text{ID-tuple}_i)$: the challenger responds as in **Phase 1**.

We refer the readers to papers by Boneh and Franklin [10] and Boneh and Silverberg [11] for examples and discussions of groups that admit such pairings.

Bilinear Diffie-Hellman (BDH) Parameter Generator: As in IBE [10] 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 admissible paring $\hat{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

BDH Problem: As in IBE [10] 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 the 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 [10] scheme, we say a BDH parameter generator $\mathcal{IG}$ satisfies the BDH assumption if the following probability is negligible in k for all PPT algorithm $\mathcal{A}$: $\Pr\left[\, \mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP, cP) = \hat{e}(P, P)^{abc} \,\right]$ where $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}) \leftarrow \mathcal{IG}(1^k)$; $P \leftarrow \mathbb{G}_1$; $a, b, c \leftarrow \mathbb{Z}q$.

3.2. fs-HIBE: Implementation

For simplicity of description, our fs-HIBE construction makes use of a version of fs-PKE scheme due to Katz [25]. In Katz's scheme, time periods are associated with the *leaf* nodes of a binary tree (Rather than with all tree nodes as in the scheme by Canetti *et al.* [12]. Our fs-HIBE scheme can also be realized based on the fs-PKE scheme by Canetti *et al.*, which will give faster key update time. The complexity discussion of our scheme is in Section 6). Without loss of generality, we give the root PKG ID_1, where ID_1 can just be an empty string.

Keys: There are two types of keys: $sk_{w,(\mathrm{ID}_1,\ldots,\mathrm{ID}_h)}$ and $SK_{i,(\mathrm{ID}_1,\ldots,\mathrm{ID}_h)}$. The node key $sk_{w,(\mathrm{ID}_1,\ldots,\mathrm{ID}_h)}$ is the key associated with some prefix w of the bit representation of a time period i and a tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_h)$. $SK_{i,(\mathrm{ID}_1,\ldots,\mathrm{ID}_h)}$ denotes the key associated with time i and an ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_h)$. It consists of sk keys as follows: $SK_{i,(\mathrm{ID}_1,\ldots,\mathrm{ID}_h)} = \{sk_{i,(\mathrm{ID}_1,\ldots,\mathrm{ID}_h)}, sk_{w1,(\mathrm{ID}_1,\ldots,\mathrm{ID}_h)}: w0 \text{ is a prefix of } i\}$. When this causes no confusion, we denote the keys as $sk_{w,h}$ and $SK_{i,h}$, respectively.

fs-HIBE construction Let $\mathcal{IG}$ be a BDH parameter generator for which the BDH assumption holds.

Setup$(1^k, N = 2^l)$: The root PKG with ID_1 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: $\{0,1\}^* \to \mathbb{G}_1$. Choose a cryptographic hash function H_2: $\mathbb{G}_2 \to \{0,1\}^n$ for some n. The security analysis will treat H_1 and H_2 as random oracles [4]. The message space is $\mathcal{M} = \{0,1\}^n$. The ciphertext space is $\mathcal{C} = \mathbb{G}_1^{l\times h} \times \{0,1\}^n$ where h is the level of the recipient. The system parameters are *params* = $(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q, H_1, H_2)$. All operations of fs-HIBE are performed under *params*. The master key is $s_\epsilon \in \mathbb{Z}q$.

   The root PKG needs to generate not only the sk key associated with the current time period 0, but also the sk keys corresponding to the internal nodes on the binary tree whose bit representations are all 0 except the last bit. The sk key for time 0 is denoted as $sk_{0^l,1}$. The rest of sk values are used by the root PKG to generate keys for future time periods, and are represented as $\{sk_{1,0}, sk_{(01),1}, \ldots, sk_{(0^{l-1}1),1}\}$. These values are generated recursively as follows.

(a) Set the secret point $S_{0,1}$ to $s_\epsilon H_1(0 \circ \mathrm{ID}_1)$, and $S_{1,1}$ to $s_\epsilon H_1(1 \circ \mathrm{ID}_1)$.
(b) Set secret key $sk_{0,1} = (S_{0,1}, \emptyset)$ and $sk_{1,1} = (S_{1,1}, \emptyset)$. Root PKG uses $sk_{0,1}$ to recursively call algorithm CompNext (defined below) to generate its secret keys. Let $(sk_{w00,1}, sk_{w01,1}) = \mathsf{CompNext}(sk_{w0,1}, w0, \mathrm{ID}_1)$, for all $1 \le |w0| \le l-1$.
(c) Set the root PKG's secret key for time period 0 as $SK_{0,1} = (sk_{0^l,1}, \{sk_{1,1}, sk_{(01),1}, \ldots, sk_{(0^{l-1}1),1}\})$, and erase all other information.

$\mathsf{CompNext}(sk_{w,h}, w, (\mathrm{ID}_1 \ldots \mathrm{ID}_h))$: This is a helper method and is called by the Setup and Upd algorithms. It takes a secret key $sk_{w,h}$, a node w, and an ID-tuple, and outputs keys $sk_{(w0),h}, sk_{(w1),h}$ for time nodes $w0$ and $w1$ of $(\mathrm{ID}_1 \ldots \mathrm{ID}_h)$.

1. Parse w as $w_1 \ldots w_d$, where $|w| = d$. Parse ID-tuple as $\mathrm{ID}_1, \ldots, \mathrm{ID}_h$. 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 $1 \le k \le d$ and $1 \le j \le h$, except for $k = 1$ and $j = 1$.
2. Choose random $s_{(d+1),j} \in \mathbb{Z}q$ for all $1 \le j \le h$.
3. Set $S_{(w0),h} = S_{w,h} + \sum_{j=1}^{h} s_{(d+1),j} H_1(w0 \circ \mathrm{ID}_1 \ldots \mathrm{ID}_j)$.
4. Set $S_{(w1),h} = S_{w,h} + \sum_{j=1}^{h} s_{(d+1),j} H_1(w1 \circ \mathrm{ID}_1 \ldots \mathrm{ID}_j)$.
5. Set $Q_{(d+1),j} = s_{(d+1),j} P$ for all $j \in 1, h$.
6. Set $\mathcal{Q}_{(w0),h}$ and $\mathcal{Q}_{(w1),h}$ to be the union of $\mathcal{Q}_{w,h}$ and $Q_{(d+1),j}$ for all $1 \le j \le h$.
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 $s_{(d+1),j}$ for all $1 \le j \le h$.

$\mathsf{KeyDer}(SK_{i,(h-1)}, i, (\mathrm{ID}_1 \ldots \mathrm{ID}_h))$: Let E_h be an 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 its key $SK_{i,(h-1)}$ as follows:

1. Parse i as $i_1 \ldots i_l$ where $l = \log_2 N$. Parse $SK_{i,(h-1)}$ 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 key $sk_{w,h}$:

   (a) Parse w as $w_1 \ldots w_d$, where $d \le l$, 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 $1 \le k \le d$. Recall that $s_{k,j}$ is a shorthand for $s_{w|_k,(\mathrm{ID}_1 \ldots \mathrm{ID}_j)}$ associated with time node $w|_k$ and tuple $(\mathrm{ID}_1 \ldots \mathrm{ID}_j)$.
   (c) Set the child entity E_h's secret point
   $S_{w,h} = S_{w,(h-1)} + \sum_{k=1}^{d} s_{k,h} H_1(w|_k \circ \mathrm{ID}_1 \ldots \mathrm{ID}_h)$.
   (d) Set $Q_{k,h} = s_{k,h} P$ for all $1 \le k \le d$. Let $\mathcal{Q}_{w,h}$ be the union of $\mathcal{Q}_{w,(h-1)}$ and $Q_{k,h}$ for all $1 \le k \le d$.
   (e) Set $sk_{w,h}$ to be $(S_{w,h}, \mathcal{Q}_{w,h})$.

3. E_h's parent sets E_h's $SK_{i,h} = (sk_{i,h}, \{sk_{(i|_{k-1}1),h}\}_{i_k=0})$, and erases all other information.

$\mathsf{Upd}(SK_{i,h}, i+1, (\mathrm{ID}_1 \ldots \mathrm{ID}_h))$ (where $i < N-1$): At the end of time i, an entity (PKG or individual) with ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_h)$ does the following to compute its private key for time $i+1$, as in the fs-PKE scheme [12,25].

1. Parse i as $i_1 \ldots i_l$, where $|i| = l$. Parse $SK_{i,h}$ as $(sk_{(i|_l),h}, \{sk_{(i|_{k-1}1),h}\}_{i_k=0})$. Erase $sk_{i|_l,h}$.
2. We distinguish two cases. If $i_l = 0$, simply output the remaining keys as the key $SK_{(i+1),h}$ for the next period for ID-tuple $(\mathrm{ID}_1, \ldots, \mathrm{ID}_h)$. Otherwise, let $\tilde{k}$ be the largest value such that $i_{\tilde{k}} = 0$ (such $\tilde{k}$ must exist since $i < N-1$). Let $i' = i|_{\tilde{k}-1}1$. Using $sk_{i',h}$ (which is included as part of $SK_{i,h}$), recursively apply algorithm $\mathsf{CompNext}$ to generate keys $sk_{(i'0^d1),h}$ for all $0 \le d \le l - \tilde{k} - 1$, and $sk_{(i'0^{d-\tilde{k}},h)}$. The key $sk_{(i'0^{d-\tilde{k}},h)}$ will be used for decryption in the next time period $i+1$; the rest of sk keys are for computing future keys. Erase $sk_{i',h}$ and output the remaining keys as $SK_{(i+1),h}$.

$\mathsf{Enc}(i, (\mathrm{ID}_1, \ldots, \mathrm{ID}_h), M)$ (where $M \in \{0,1\}^n$):

1. Parse i as $i_1 \ldots i_l$. Select random $r \leftarrow \mathbb{Z}q$.
2. Denote $P_{k,j} = H_1(i|_k \circ \mathrm{ID}_1 \ldots \mathrm{ID}_j)$ for all $1 \le k \le l$ and $1 \le j \le h$. Output $\langle i, (\mathrm{ID}_1, \ldots, \mathrm{ID}_h), C\rangle$, where $C = (rP, rP_{2,1}, \ldots, rP_{l,1}, rP_{1,2}, \ldots, rP_{l,2}, \ldots, rP_{1,h}, \ldots, rP_{l,h}, M \oplus H_2(\hat{e}(Q, H_1(i_1 \circ \mathrm{ID}_1))^r))$.

$\mathsf{Dec}(i, (\mathrm{ID}_1, \ldots, \mathrm{ID}_h), SK_{i,h}, C)$:

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

Using Fujisaki-Okamoto padding [17] and the help of random oracles H_3 and H_4, algorithm Enc and Dec can be converted into ones with chosen ciphertext security, as in BF-IBE [10] and GS-HIBE [20]. We summarize the security of our fs-HIBE scheme in Theorem 3.1 and 3.2.

Theorem 3.1. *Suppose there is a nonadaptive adversary $\mathcal{A}$ that has advantage ϵ against the one-way secure fs-HIBE scheme for some fixed time t and ID-tuple, and that makes $q_{H_2} > 0$ hash queries to the hash function H_2 and a finite number of key derivation 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 $\mathcal{O}(time(\mathcal{A}))$.*

Theorem 3.2. *Suppose there is an adaptive adversary $\mathcal{A}$ that has advantage ϵ against the one-way secure fs-HIBE scheme targeting some time and some ID-tuple at level h, and that makes $q_{H_2} > 0$ hash queries to the hash function H_2 and at most $q_E > 0$ key derivation queries. Let $l = \log_2 N$, where N is the total number of time periods. 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 $(\epsilon(\frac{h+l}{e(2lq_E+h+l)})^{(h+l)/2} - \frac{1}{2^n})/q_{H_2}$ and running time $\mathcal{O}(time(\mathcal{A}))$.*

4. Application: Forward-Secure Broadcast Encryption

In this section, we show how the fs-HIBE scheme can be used to build a scalable forward-secure public-key broadcast encryption (fs-BE) scheme which is joining-time-oblivious. In what follows, N denotes the total number of time periods, $\mathcal{E}$ denotes the universe of users and $E = |\mathcal{E}|$.

4.1. fs-BE: Syntax

Forward-Secure Broadcast Encryption Scheme (fs-BE): An fs-BE scheme is specified by five poly-time algorithms Setup, KeyDer, Upd, Enc, Dec:

Setup: The setup algorithm is a probabilistic algorithm run by the center to set up the parameters of the scheme. Setup takes as input a security parameter k and possibly $r_{\max}$ (where $r_{\max}$ is a revocation threshold, i.e. the maximum number of users that can be revoked). The input also includes the total number E of users in the system and the total number of time periods N. Setup generates the public key PK and the initial master secret key MSK_0.

KeyDer: The key derivation algorithm is a probabilistic algorithm run by the center to compute the secret initialization data for a new user. KeyDer takes as input the master secret key MSK_t at time t, the identity u of the user and the current time period $t < N-1$ and outputs the new secret key $USK_{t,u}$.

Upd: The key update algorithm is a deterministic algorithm run by an entity (center or user) to update its own secret key of time t into a new secret key valid for the following time period $t+1$. For a user, Upd takes as input the public key PK, the identity u of a user, the current time period $t < N-1$, and the user's secret key $USK_{t,u}$, and outputs the new user's secret key $USK_{t+1,u}$. For the center, the algorithm takes as input the public key PK, the current time period $t < N$, and the key MSK_t, and outputs the secret key MSK_{t+1}.

Enc: The encryption algorithm is a probabilistic algorithm that each content provider can use to encrypt messages. Enc takes as input the public key PK, a message M, the current time period t and a set $\mathcal{R}$ of revoked users (with $|\mathcal{R}| \leq r_{\max}$, if a threshold has been specified to the Setup algorithm), and returns the ciphertext C to be broadcast.

Dec: The decryption algorithm is a deterministic algorithm run by each user to recover the content from the broadcast. Dec takes as input the public key PK, the identity u of a user, a time period $t < N$, the user's secret key $USK_{t,u}$ and a ciphertext C, and returns a message M.

An fs-BE scheme should satisfy the following correctness constraint: for any pair (PK, MSK_t) output by the algorithm $\mathsf{Setup}(k, r_{\max}, N, E)$, any $t < N$, any $\mathcal{R} \subseteq \mathcal{E}$, $(|\mathcal{R}| \leq r_{\max})$, any user $u \in \mathcal{E} \setminus \mathcal{R}$ with secret key $USK_{t,u}$ (properly generated for time period t) and any message M, it should hold that:

$$M = \mathsf{Dec}(PK, u, t, USK_{t,u}, \mathsf{Enc}(PK, M, t, \mathcal{R})).$$

4.2. fs-BE: Security

In fs-BE scheme, if a user leaks his or her secret key and is not revoked by a content provider, the security of subsequent communications broadcasted by such provider is compromised. As a matter of fact, the forward security of broadcast encryption schemes guarantees that this is the *only* case where unauthorized access to the broadcast content may occur. The advantage of the adversary is not significantly improved even if she corrupts multiple users at different time periods. We formalize the security definition of fs-BE below.

Chosen-ciphertext Security: An fs-BE scheme is *forward-secure against chosen-ciphertext 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 security parameters $k, r_{\max}$, and runs the Setup algorithm, for the specified number of users E and time periods N. It gives the adversary the resulting system public key PK and keeps the initial master secret key MSK_0 secret to itself.

Phase 1: The adversary issues, in any adaptively-chosen order, queries $q_1, \ldots, q_m$, where q_i is one of the followings:

1. Key derivation query (u, t): the challenger runs algorithm $\mathsf{KeyDer}(MSK_t, u, t)$ to generate the private key $USK_{t,u}$ corresponding to user u at time t, and sends $USK_{t,u}$ to the adversary.
2. Decryption query (u, t, C): the challenger first runs the $\mathsf{KeyDer}(MSK_t, u, t)$ algorithm to recover private key $USK_{t,u}$ corresponding to user u at time t, and then runs decryption algorithm $\mathsf{Dec}(PK, u, t, USK_{t,u}, C)$ to decrypt C, and sends the resulting plaintext to the adversary.

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^* on which it wishes to be challenged. The challenger picks a random bit $b \in \{0, 1\}$, and set $C^* = \mathsf{Enc}(PK, M_b, t^*, \mathcal{R}_{t^*})$, where $\mathcal{R}_{t^*} = \{u \mid \mathcal{A}$ asked a key derivation query for (u, t), for some $t \leq t^*\}$. 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. Key derivation query (u, t): the challenger first checks that either $u \in \mathcal{R}_{t^*}$ or $t > t^*$ and if so, it responds as in **Phase 1**. Notice that if a bound r_{max} was specified in Setup, then the adversary is restricted to corrupt at most r_{max} distinct users via key derivation queries.
2. Decryption query (u, t, C): the challenger first checks that either $C \neq C^*$ or $u \in \mathcal{R}_{t^*}$ or $t \neq t^*$ and if so, it responds as in **Phase 1**.

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

4.3. fs-BE: A Construction Based on fs-HIBE

Here, we show how our fs-HIBE scheme can be applied to the construction of the public-key broadcast encryption of [14] to obtain a forward-secure public-key BE scheme. Dodis and Fazio [14] provided a construction that extends the symmetric-key broadcast encryption scheme of Naor *et al.* [29] to the public-key setting, based on any secure HIBE scheme. The construction of [14] also applies to the scheme of Halevy and Shamir [22], that improves upon the work of [29]. The symmetric-key BE scheme of Halevy and Shamir is an instance of the *Subset Cover Framework* [29]. The main idea of the framework is to define a family $\mathcal{S}$ of subsets of the universe $\mathcal{E}$ of users in the system, and to associate each subset with a key, which is made available to all the users belonging to the given subset. To broadcast a message to all the subscribers except those in some set $\mathcal{R}$, a content provider first *covers* the set of privileged users using subsets from the family $\mathcal{S}$. This is done by identifying a partition of $\mathcal{E} \setminus \mathcal{R}$, where all the subsets are elements of $\mathcal{S}$. Then, the provider encrypts the message for all the subsets in that partition. To decrypt, a user $u \notin \mathcal{R}$ first identifies the subset in the partition of $\mathcal{E} \setminus \mathcal{R}$ to which he belongs, and then recovers the corresponding secret keys from his secret information.

In the public-key BE scheme [14], the subsets containing a given user are organized into groups, and a special secret key, *protokey*, is associated with each of these groups. A user only needs to store these protokeys, from which he can derive the actual decryption keys corresponding to all the subsets in the group. Such an organization of the subsets of the family $\mathcal{S}$ produces a hierarchy, in which the leaves are elements of $\mathcal{S}$ and each internal node corresponds to a group of subsets. Using HIBE, a secret key can be associated with each internal node in the hierarchy, and constitutes the protokey for the group corresponding to that internal node.

In order to add forward secrecy in the public-key BE scheme, we essentially apply the fs-HIBE scheme to the above hierarchy. In fs-BE scheme, a protokey is associated with not only a node in the hierarchy, but also with a time period t. In fs-BE Setup, the center runs fs-HIBE Setup algorithm to compute its master secret $SK_{0,1}$. This key evolves with time, and is used by the center to compute protokeys for users. In fs-BE KeyDer, a user joins the broadcast at some time t, and the center uses its current master secret key $SK_{t,1}$ to derive protokeys for the user by running fs-HIBE KeyDer algorithm. The center and users evolve their secret keys with time autonomously by calling algorithm Upd of fs-HIBE. In fs-BE Enc, a content provider uses fs-HIBE Enc algorithm to encrypt the message not only with respect to the nodes in the hierarchy that represents the subsets in the partition of $\mathcal{E} \setminus \mathcal{R}$, but also to the current time t. In fs-BE Dec, the user first runs fs-HIBE KeyDer to derive the current secret keys from his protokey at time t. These secret keys are used for decryption by runing fs-HIBE Dec algorithm. The detailed construction of our fs-BE scheme is ommitted here. We analyze the complexity of fs-BE operations in Section 6.

5. Application: Multiple Hierarchical Identity-Based Encryption Scheme

ID-based cryptographic schemes have been used in complex access control scenarios [23, 33]. We generalize the fs-HIBE into a collusion resistant multiple hierarchical ID-based encryption (MHIBE) scheme, where a message can be encrypted under multiple ID-tuples. The applications of MHIBE scheme include secure communications with users having multiple identities.

Motivations for MHIBE In role-based access control systems (RBAC) [32], individuals are assigned roles according to their qualifications, and access decisions are based on roles. The use of roles to control access is proven to be an effective means for streamlining the security management process [32]. Communications to a specific role may need to be protected so that messages can be read only by members of that role. This can be done using a shared key approach, which can be realized by an HIBE scheme. Members of a role are given a secret group key that is used for decrypting messages encrypted with the group public key of that role, which is an ID-tuple in HIBE. For example, the public key of the role *doctor* in the Emergency Room at a hospital is the ID-tuple (Hospital, ER, doctor), and members of the role *doctor* are given the corresponding private key in HIBE. The hierarchical structure of public keys in HIBE makes it particularly suitable for managing role communications in large organizations. This group key approach is efficient and scalable compared to encrypting the message with individual recipients' personal public keys, because a message is encrypted only once (under the public key of the role).

A user may have multiple roles. Some messages are intended to be read only by those who have multiple roles, and should not be recovered by collusions among role members. For example, the intended message recipients are those who must take on both role *doctor* in ER and role *research manager* at the affiliated medical school of the hospital. In healthcare systems, medical data such as patient records are extremely sensitive, therefore, achieving this type of secure communications is important. However, the GS-HIBE [20] scheme provides cryptographic operations only if the message is encrypted under one identity (ID-tuple). It cannot be used for communications to an *intersection* of identities. Note that the Dual-Identity-Based Encryption scheme by Gentry and Silverberg [20] is different from what we want to achieve here. The word "dual" in their scheme [20] refers that the identities of both the sender and the recipient, rather than just the recipient, are required as input into the encryption and decryption algorithms.

To solve the problem of secure communications to members having multiple roles, we develop a multiple hierarchical identity-based encryption (MHIBE) scheme, where encryption is under multiple ID-tuples. In addition, it can be used for authenticating multiple hierarchical identities in the hidden credential protocol [23], where the success of authentication of identities is implied if one can correctly decrypt the message encrypted with the required identities of the intended recipients. What makes the problem interesting is that the *intersection* of identities is different from the *union* of identities, which implies that a proper scheme should be collusion-resistant: secure even if adversaries with partial roles collude. In other words, it requires that compromising the private keys of individual identities does not compromise the messages encrypted with the intersection of identities. This property cannot be achieved by the broken Scheme III described in Section 1.3, where two separate HIBE schemes are used, as it is not collusion-resistant.

Next we use an example to describe the MHIBE scheme, including key acquisition, encryption, and the properties of MHIBE implementation generalized from our fs-HIBE scheme.

5.1. Identity-set and Joining-path-obliviousness

In MHIBE, we define an *identity-set* as the set of identities that a user has, each represented as an ID-tuple. For example, Bob's identity-set is {(Hospital, ER, Doctor), (Hospital, School, Manager)}. An *ancestor* E' of a node E has the same number of ID-tuples in its identity-set as that of E, and for each ID-tuple T in the identity-set of E, there is an ID-tuple in the identity-set of E' such that it is either the ancestor of T in HIBE or the same as T. In addition, the ancestor E' of the node E cannot be E. All ancestors of node E are capable of generating secret keys for E.

In an MHIBE scheme, Bob may obtain his key directly from either of the two ancestor entities. One is the entity whose identity-set is {(Hospital, ER), (Hospital, School, Manager)}. And the other has the identity-set {(Hospital, ER, Doctor), (Hospital, School)}. Bob's parents obtain their keys from their parents in the same way. The highest-level ancestor in this example is the hospital and has the identity-set {Hospital, Hospital} (not {Hospital}). The root secret s_ϵ used for computing the private key for identity-set {Hospital, Hospital} may be the same as the root secret used in regular HIBE scheme [20]. The private key is set to $s_\epsilon H_1(\text{Hospital} \circ \text{Hospital})$. Bob's key can be computed only by his ancestors in the MHIBE scheme. An MHIBE scheme needs to be *joining-path-oblivious*. This means that encryption should be oblivious of the path from which the receiver and his ancestors acquire their private keys. Having the receiver's identity-set is sufficient to encrypt a message. For example, the sender does not need to know whether Bob obtains his keys from entity {(Hospital, ER), (Hospital, School, Manager)} or from entity {(Hospital, ER, Doctor), (Hospital, School)}.

5.2. Properties of Our MHIBE Implementation

Our fs-HIBE scheme naturally gives rise to an MHIBE scheme. In fs-HIBE, a message is encrypted under both an ID-tuple and the current time. This can be viewed as the encryption under two tuples, one being the current time. Therefore, the identities in MHIBE scheme capture a broader sense of meaning. The MHIBE scheme generalized from our fs-HIBE scheme supports dynamic joins and joining-path-oblivious encryption. More importantly, it is collusion-resistant, which cannot be achieved by using multiple separate HIBE [20] schemes. In our MHIBE implementation, a message encrypted under {(Hospital, ER, Doctor), (Hospital, School, Manager)} or {(Hospital, School, Manager), (Hospital, ER, Doctor)} requires different decryption keys. We note that in this scheme, the fact that a user holds the private key corresponding to multiple identities does not imply that he or she has the private key to any subset of identities.

Our MHIBE scheme has similar goals as the pairing-based attribute-based encryption (ABE) schemes [31]. In ABE, a user's private key for an application is constructed so that the key can encode expressive access control policies. While ABE can support expressive policies, a user may have to store several private keys, each for one application/policy. In comparison, MHIBE does not support general access control policies; the private keys in MHIBE are generated independent of applications or policies. Whether or not MHIBE can be realized by ABE is an interesting open question. We omit the details of MHIBE scheme (definition of security, description of scheme, and proof of security), as this is a direct generalization of fs-HIBE scheme. The complexities of various parameters of our MHIBE scheme are shown in Table 1 in Section 6.

6. Discussions

We analyze the complexity of our fs-HIBE scheme, the generalized MHIBE scheme, and the fs-BE scheme in Table 1 showing running time complexities and key sizes. Key generation time of fs-HIBE and MHIBE is the time to generate secret keys for a child node by the parent. Key generation time of fs-BE scheme is the running time of fs-BE KeyDer algorithm. In our fs-HIBE scheme, the time periods correspond to leaf nodes of a binary tree, and the key update time is $\mathcal{O}(h \log N)$, where N is the total number of time periods and h is the length of an ID-tuple. Because of the node arrangement, the key generation time and key update time of our fs-HIBE scheme grows logarithmically with the total number of time periods N. Faster key update time ($\mathcal{O}(h)$) can be achieved, if the time periods are associated with *all* the nodes of the tree in a pre-order traversal, as in the fs-PKE scheme by Canetti *et al.* [12]. Because the realization of such a fs-HIBE scheme can be easily derived from the construction in Section 3.2, it is omitted here. We show the optimized running time in Table 1. Even dropping the joining-time-obliviousness requirement (as in the naive Scheme II of Section 1.3), our implementation cannot achieve a ciphertext with linear length $\mathcal{O}(h + \log N)$.

Table 1. Dependency of parameters of our fs-HIBE, MHIBE, and fs-BE schemes on the total number N of time periods, the length h of an ID-tuple, the number m of ID-tuples in an identity-set in MHIBE, the total number E of fs-BE users and the number r of actual revoked users in fs-BE scheme. Key derivation time of fs-HIBE and MHIBE is the time to generate secret keys for a child node by the parent. Key derivation time of fs-BE scheme is the running time of fs-BE KeyDer algorithm.

Parameters	fs-HIBE	MHIBE	fs-BE
Key derivation time	$\mathcal{O}(h \log N)$	$\mathcal{O}(h^m)$	$\mathcal{O}(\log^3 E \log N)$
Encryption time	$\mathcal{O}(h \log N)$	$\mathcal{O}(h^m)$	$\mathcal{O}(r \log E \log N)$
Decryption time	$\mathcal{O}(h \log N)$	$\mathcal{O}(h^m)$	$\mathcal{O}(r + \log E \log N))$
Key update time	$\mathcal{O}(h)$	N/A	$\mathcal{O}(\log^3 E)$
Ciphertext length	$\mathcal{O}(h \log N)$	$\mathcal{O}(h^m)$	$\mathcal{O}(r \log E \log N)$
Public key size	$\mathcal{O}(h + \log N)$	$\mathcal{O}(hm)$	$\mathcal{O}(r \log E + \log N)$
Secret key size	$\mathcal{O}(h \log N)$	$\mathcal{O}(h^m)$	$\mathcal{O}(\log^3 E \log N)$

7. Conclusion

The Multiple Hierarchical Identity-Based Encryption scheme is an ID-Based encryption scheme for complex hierarchies. The generalization of a collusion-resistant MHIBE scheme from the Hierarchical Identity-Based Encryption scheme is significant, because MHIBE scheme conveniently lends itself to a wide range of applications that cannot be accomplished using HIBE schemes. To demonstrate this, we presented in details a forward-secure HIBE scheme and a forward-secure Broadcast Encryption scheme. We also described the application of MHIBE in the access control paradigm. The forward-secure applications derived from our MHIBE scheme are joining-time-oblivious and support dynamic joins, which make them scalable.

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*, pages 441–456. Springer-Verlag, 2001.

[2] R. Anderson. Two remarks on public-key cryptology. Invited lecture, *4th ACM Conference on Computer and Communications Security*, 1997. Available at `http://www.cl.cam.ac.uk/ftp/users/rja14/`.

[3] M. Bellare and S. K. Miner. A forward-secure digital signature scheme. In *Advances in Cryptology — Crypto '99*, volume 1666 of *LNCS*, pages 431–448. Springer-Verlag, 1999.

[4] 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.

[5] M. Bellare and B. Yee. Forward security in private-key cryptography. In *CT-RSA*, volume 2612 of *LNCS*, pages 1–18. Springer-Verlag, 2003.

[6] S. Berkovits. How to broadcast a secret. In *Advances in Cryptology — Eurocrypt '91*, volume 547 of *LNCS*, pages 535–541. Springer-Verlag, 1991.

[7] D. Boneh and X. Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *Advances in Cryptology — Eurocrypt '04*, volume 3027 of *LNCS*, pages 223–238. Springer-Verlag, 2004.

[8] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology — Crypto '04*, volume 3152 of *Lecture Notes in Computer Science*, 2004.

[9] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Proceedings of Advances in cryptology — Eurocrypt '05*, Lecture Notes in Computer Science, 2005.

[10] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology — Crypto '01*, volume 2139 of *LNCS*, pages 213–229. Springer-Verlag, 2001.

[11] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2003.

[12] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology — Eurocrypt '03*, volume 2656 of *LNCS*, pages 255–271. Springer-Verlag, 2003.

[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] Y. Dodis and N. Fazio. Public-key broadcast encryption for stateless receivers. In *Digital Rights Management — DRM '02*, volume 2696 of *LNCS*, pages 61–80. Springer, 2002.

[15] Y. Dodis and N. Fazio. Public-key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *Public Key Cryptography — PKC '03*, volume 2567 of *LNCS*, pages 100–115. Springer-Verlag, 2003.

[16] A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology — Crypto '93*, volume 773 of *LNCS*, pages 480–491. Springer-Verlag, 1993.

[17] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology — Crypto '99*, volume 1666 of *LNCS*, pages 537–554. Springer-Verlag, 1999.

[18] A. Garay, J. Staddon, and A. Wool. Long-lived broadcast encryption. In *Advances in Cryptology — Crypto 2000*, volume 1880 of *LNCS*, pages 333–352. Springer-Verlag, 2000.

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

[20] C. Gentry and A. Silverberg. Hierarchical ID-based cryptography. In *Advances in Cryptology — Asiacrypt '02*, volume 2501 of *LNCS*, pages 548–566. Springer-Verlag, 2002.

[21] M. T. Goodrich, J. Z. Sun, and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. In *Advances in Cryptology — Crypto '04*, LNCS, 2004.

[22] D. Halevy and A. Shamir. The LSD broadcast encryption scheme. In *Advances in Cryptology — Crypto '02*, volume 2442 of *LNCS*, pages 47–60. Springer-Verlag, 2002.

[23] J. Holt, R. Bradshaw, K. E. Seamons, and H. Orman. Hidden credentials. In *Proceedings of the 2nd ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 1–8, October 2003.

[24] J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In *Advances in Cryptology — Eurocrypt '02*, volume 2332 of *LNCS*, pages 466–481. Springer-Verlag, 2002.

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

[26] C. Kim, Y. Hwang, and P. Lee. An efficient public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *Advances in Cryptology — Asiacrypt 2003*, volume 2894 of *LNCS*, pages 359–373. Springer-Verlag, 2003.

[27] M. Luby and J. Staddon. Combinatorial bounds for broadcast encryption. In *Advances in Cryptology — Eurocrypt '98*, volume 1403 of *LNCS*, pages 512–526. Springer-Verlag, 1998.

[28] 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*, pages 400–417. Springer-Verlag, 2002.

[29] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology — Crypto '01*, volume 2139 of *LNCS*, pages 41–62. Springer-Verlag, 2001.

[30] M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *Financial Cryptography — FC '00*, volume 1962 of *LNCS*, pages 1–20. Springer-Verlag, 2000.

[31] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security (CCS)*, pages 195–203, 2007.

[32] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29, Number 2:38–47, 1996.

[33] R. Tamassia, D. Yao, and W. H. Winsborough. Role-based cascaded delegation. In *Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT '04)*, pages 146 – 155. ACM Press, June 2004.

[34] W. Tzeng and Z. Tzeng. A public-key traitor tracing scheme with revocation using dynamics shares. In *Public Key Cryptography — PKC '01*, volume 1992 of *LNCS*, pages 207–224. Springer-Verlag, 2001.

[35] B. R. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology — Eurocrypt '05*, Lecture Notes in Computer Science, 2005.

[36] C. Wong, M. Gouda, and S. Lam. Secure group communications using key graphs. In *Proceedings of the ACM SIGCOMM '98*, pages 68 – 79. ACM Press, 1998.

[37] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption. Full paper at `http://www.cs.brown.edu/people/dyao/fs-hibe-full.pdf`.

[38] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 354 – 363. ACM Press, 2004.