Adaptive Key Protection in Complex Cryptosystems with Attributes

Zilong Wang*
School of Mathematical Sciences,
Peking University, Beijing 100871, P.R. China
zlwangmath@gmail.com

Danfeng (Daphne) Yao†
Department of Computer Science,
Virginia Tech, Blacksburg VA 24060, USA
danfeng@cs.vt.edu

Rongquan Feng
School of Mathematical Sciences,
Peking University, Beijing 100871, P.R. China
fengrq@math.pku.edu.cn

Abstract

In the attribute-based encryption (ABE) model, attributes (as opposed to identities) are used to encrypt messages, and all the receivers with qualifying attributes can decrypt the ciphertext. However, compromised attribute keys may affect the communications of many users who share the same access control policies. We present the notion of forward-secure attribute-based encryption (fs-ABE) and give a concrete construction based on bilinear map and decisional bilinear Diffie-Hellman assumption. Forward security means that a compromised private key by an adversary at time t does not break the confidentiality of the communication that took place prior to t. We describe how to achieve both forward security and encryption with attributes, and formally prove our security against the adaptive chosen-ciphertext adversaries. Our scheme is non-trivial, and the key size only grows

*This work was done by the first author when he was a visitor at Virginia Tech. He was supported in part by Graduate School of Peking University (2010-02-020).

†This work has been supported in part by NSF grant CAREER CNS-0953638 and CNS-0831186.

polynomially with $\log N$ (where N is the number of time periods). We further generalize our scheme to support the individualized key-updating schedule for each attribute, which provides a finer granularity for key management. Our insights on the required properties that an ABE scheme needs to possess in order to be forward-secure compatible are useful beyond the specific fs-ABE construction given. We raise an open question at the end of the paper on the escrow problem of the master key in ABE schemes.
Keywords: Attribute-based encryption, forward security, key update

1 Introduction

The main feature of a forward-secure encryption scheme is that the compromise of current decryption keys does not compromise past decryption keys; therefore the confidentiality of past communications is preserved. Forward security aims at mitigating the damages caused by stolen secret keys. This concept was first coined by Günther [1] and later by Diffie et al. [2]. In forward secure schemes, 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 for any *prior* time period. To prevent the adversary from breaking the security of the system for any *subsequent* time period, one needs to revoke the compromised keys. Solutions for supporting the forward security have been proposed in the context of symmetric-key encryption schemes [3], public-key encryption schemes [4], identity-based encryption [5], digital signature schemes [6, 7, 8], and recently in cloud-based content delivery [9].

In this paper, we point out that forward security is important for preserving the security of attribute-based encryption (ABE) schemes [10, 11, 12]. In ABE schemes, attributes (as opposed to identities) are used for encryption, and only users with qualified attributes (and corresponding private keys) can decrypt. Attribute-based encryption enables the implicit encoding of authorization policies in the private key of the user (e.g., key-policy attribute-based encryption [10]) or in the ciphertext (e.g., ciphertext-policy attribute-based encryption [13]). However, because the attribute-based secrets are *shared* among all qualifying users, compromised keys can affect other users' communication. This *shared key* feature in the key management of ABE motivates our work on designing *forward-secure attribute-based encryption* (fs-ABE). In a fs-ABE, any compromised attribute key at time t does not affect the confidentiality of any of the prior communication encrypted with that attribute.

Attribute-based encryption has practical applications in advanced access control and data management, such as electronic medical records [14, 15]. In real world, ABE system can be naturally used to construct a targeted broadcast system, which is first descibed in [10]. In this kind of system, the needs of individual users is targeted at, while a broadcast channel still offer the content with economies-of-scale. For such an application, the construction based on ABE scheme is more efficent than other broadcast encryption schemes. As in broadcast encryption, forward security in this situation is very important. Forward secure ABE schemes can improve the practicability of the contruction.

Our contributions are summarized as follows.

- We give the model and definition for a general forward-secure attribute-based encryption scheme that is secure against adaptive chosen-ciphertext adversaries. We define the security in a game model allowing the adversary to issue key-generation queries and decryption queries. The users refresh their private keys *autonomously*, which is scalable. We support *dynamic join* where users can join the fs-ABE system at any time.

- We give a concrete instantiation of a forward-secure attribute-based encryption scheme, where secrets corresponding to all the attributes evolve and update based on the same schedule. Our construction makes use of the operations in key-policy attribute-based encryption scheme by Goyal, Pandey, Sahai, and Waters (GPSW-ABE) [10], and inherits its bilinear map usage as well as the hardness assumption of decisional bilinear Diffie Hellman (DBDH). We formally prove the security of our scheme and analyze the complexities of the operations.

  We also provide insights on the general properties that an ABE scheme should posses in order to support forward security. We explain why CP-ABE scheme cannot be converted into a non-trivial forward secure one with the existing cryptographic tools and leave it as an open problem.

- We further improve the flexibility of our basic fs-ABE scheme by supporting the individualized key-updating schedule for each attribute, and provide the sketch for its security. Individualized key update improves the efficiency, usability, and security of the fs-ABE scheme.

Our work provides the insights and solutions to the problem of hardening authorization-enabling encryption schemes namely attribute-based encryp-

tion (ABE) against compromised secret keys. Although our specific construction is based on key-policy ABE (KP-ABE), we summarize the properties that can be used determine whether an attribute-based encryption scheme is forward-secure compatible or not. This contribution is significant beyond the specific KP-ABE scheme studied.

Organization of the paper Related work is given in the next section. We present our definitions and model of a forward-secure attribute-based encryption scheme in Section 3. We give some trivial schemes and discuss why they are not good in Section 4. We present requirements that an ABE scheme needs to satisfy in order to become foward-secure compatible in Section 5. A concrete construction based on bilinear maps is described in Section 6. Our basic fs-ABE scheme is extended into a ifs-ABE scheme in 7. Several extensions are presented to further improve the security of our scheme in Section 8. Conclusions and an open problem are given in Section 9. Our formal proof is given in Appendix B.

2 Related Work

The notion of non-interactive forward security was proposed by Anderson [16], which was formalized by Bellare and Miner [6]. Forward-secure digital signature schemes were also proposed [6, 7, 8]. Bellare and Yee [3] provided a comprehensive description of forward security in the context of symmetric-key based cryptographic primitives. The first forward-secure public-key encryption (fs-PKE) scheme was given by Canetti et al. [4], based on the decisional bilinear Diffie-Hellman assumption [17].

The fs-PKE scheme in [4] 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 data structure was inspired by the identity hierarchy in the Gentry-Silverberg HIBE scheme [18], and is also used by our work to encode the time information in policies.

Authors in [5] investigated how to bring forward security to the hierarchical identity-based encryption (HIBE) scheme by providing private keys that are both self-evolving for forward secrecy and delegatable for generating identity-based keys. Due to the inherent key-escrow property, key exposure

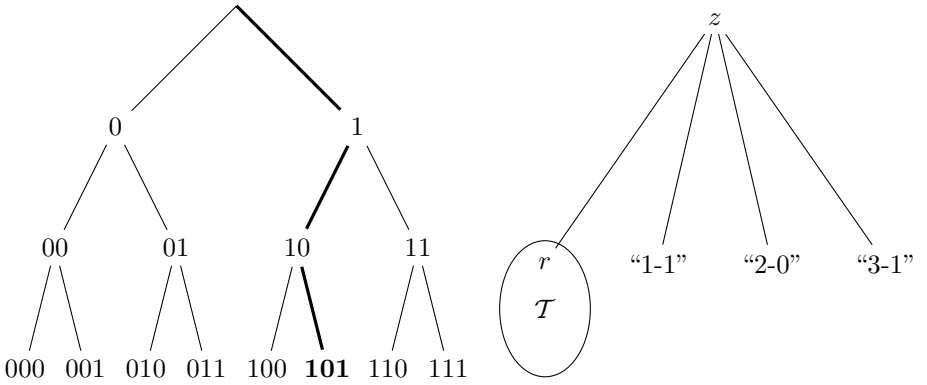
is a realistic threat over the lifetime of such a scheme, and the standard notion of HIBE security crucially depends on secret keys remaining secret. The forward-secure hierarchical identity-based encryption (fs-HIBE) scheme allows each user in the hierarchy to refresh his or her private keys periodically while keeping the public key the same.

Attribute based encryption schemes are related to the identity-based encryption (IBE) schemes [19, 20, 21, 22, 23], as both cryptosystems are novelty public-key encryption schemes with properties that can be leveraged for authorization. Solutions have been proposed to utilize IBE schemes for access control [5, 24], specifically encoding authorization policies in the public keys. Researchers recently found that ABE schemes can provide expressive and anonymous authorization mechanisms for the information sharing in the cloud [25, 26, 27, 28]. A comprehensive survey on ABE schemes can be found in [29].

3 Definitions and Models in Forward-secure Attribute-Based Encryption

We define the syntax of forward-secure ABE (fs-ABE) scheme and security model for such schemes in this section. Most of the recent ABE schemes can be categorized as key-policy attribute-based encryption (KP-ABE) or ciphertext-policy attribute-based encryption (CP-ABE). In KP-ABE schemes (e.g., [10]), each private key is associated with an access structure that is chosen by the authority, and each ciphertext is labeled with a set of attributes. A private key can decrypt a ciphertext if and only if the attributes in the ciphertext satisfy the access structure in the private key. A KP-ABE scheme can be realized with a tree access structure which is composed of threshold gates as interior nodes and the attributes as leaves. Ostrovsky et al. [11] described a KP-ABE scheme where non-monotonic access structures can be specified and enforced. Bethencourt et al. [13] gave the first concrete CP-ABE scheme, where each private key is described by a set of attributes, and each ciphertext is associated with an access structure that is determined by the party encrypting the message. Our definitions and construction in this paper follows the KP-ABE paradigm. We point out the challenges associated with providing non-trivial forward security for CP-ABE schemes in Section 5.

In our foward-secure ABE scheme, the private key of a user is evolved with time. At the time period 0 a user is issued an initial private key associated with an access tree by the Private Key Generator (PKG). At



(a) The time period $\langle 5 \rangle = 101$. (b) The tree $\mathcal{T}_{101}$.

Figure 1: The time period $\langle 5 \rangle = 101$ in (a) and the tree $\mathcal{T}_{101}$ in (b) associated with the time period $\langle 5 \rangle = 101$, where the total number of time periods is $N = 2^d = 2^3$.

the end of each time period he updates his private key for the next time period and erases the current key. During the time period i, a message is encrypted using a set of attributes and the time period i. A user can decrypt the encrypted message using his private key for the time period i, if and only if the access tree in his private key can be satisfied by the set of attributes in the ciphertext.

3.1 Notations

Time Period We assume for simplicity that the total number N of time periods is a power of 2; that is $N = 2^d$. Let $\langle i \rangle$ denote the d-bit representation of the time period i (where $0 \leq i \leq 2^d - 1$). Let $w = w_1 w_2 \cdots w_l$ be the l-bit prefix of the bit representation of some time period. In our fs-ABE scheme, time periods are associated with the leaf nodes of a binary tree. This representation of time follows that of fs-PKE [4]. Figure 1(a) shows the time period $\langle 5 \rangle = 101$, where $d = 3$. A simple tree $\mathcal{T}_{101}$ associated with the time period $\langle 5 \rangle = 101$ is shown in Figure 1(b).

Access Trees An access structure can be represented by a tree $\mathcal{T}$. Each interior node x of the tree represents a threshold gate with a positive threshold value k_x, which is not greater than the number n_x of children of the node. We denote a k_x of n_x threshold gate by (k_x, n_x)-gate. Each leaf node x of the tree represents an attribute, and the threshold value k_x associated with

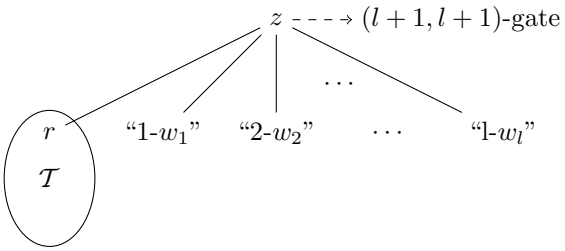


Figure 2: The tree $\mathcal{T}_w$, where "1-w_1", "2-w_2", $\cdots$, "l-w_l" are components of a prefix $w = w_1 w_2 \cdots w_l$ $(1 \leq l \leq d)$ of some time period. The time period is d-bit long.

the leaf node is defined as 1. We define the function parent(x) as the parent of the node x. If x is a leaf node, then we use att(x) to denote the attribute associated with x. The children of every node are numbered from 1 in some order. Such a number associated with a node x can be returned by the function index(x).

We denote the subtree of $\mathcal{T}$rooted at the node x by $\mathcal{T}|_x$. Thus $\mathcal{T}$ is the same as $\mathcal{T}|_r$, where r is the root node of the tree $\mathcal{T}$. If a set γ of attributes satisfies the access tree $\mathcal{T}|_x$, we denote it as $\mathcal{T}|_x(\gamma) = 1$. If x is an interior node, then $\mathcal{T}|_x$ is satisfied by γ if and only if at least k_x subtrees $\mathcal{T}|_{x'}$ of $\mathcal{T}|_x$ is satisfied, where x' is a child of x. If x is a leaf node, then $\mathcal{T}|_x(\gamma) = 1$ if and only if att$(x) \in \gamma$.

In the construction of fs-ABE scheme, we add a trivial $(1, 1)$-gate z above the root node r of an access tree $\mathcal{T}$ to obtain the tree $\mathcal{T}_\varepsilon$, then add attributes "1-w_1", "2-w_2", $\cdots$, "l-w_l" in turn to the tree $\mathcal{T}_\varepsilon$ as children of the root node z of $\mathcal{T}_\varepsilon$, where "1-w_1", "2-w_2", $\cdots$, "l-w_l" are the components of a prefix $w = w_1 w_2 \cdots w_l (1 \leq l \leq d)$ of some time period. We denote the new tree by $\mathcal{T}_w$. Notice that the root node z is converted from $(1, 1)$-gate to $(l+1, l+1)$-gate. Let $sk_{\mathcal{T},w}$ denote the secret key associated with $\mathcal{T}_w$. The tree $\mathcal{T}_w$ is illustrated in Figure 2.

Keys We denote the private key associated with an access tree $\mathcal{T}$ and a time period i by $SK_{\mathcal{T},i}$. The private key $SK_{\mathcal{T},i}$ consists of some secret keys associated with the tree $\mathcal{T}$ and some prefixes of the time period i:

$$SK_{\mathcal{T},i} = \left(sk_{\mathcal{T},\langle i\rangle}, \{sk_{\mathcal{T},i_0 i_1 \cdots i_{k-1} 1}\}_{i_k=0}\right)$$

where $sk_{\mathcal{T},\langle i\rangle}$ is the secret key used to decrypt a ciphertext, $\{sk_{\mathcal{T},i_0 i_1 \cdots i_{k-1} 1}\}_{i_k=0}$ are the secret keys used by a user to update his private key for the next time

period $i+1$. Notice that if $i_0 i_1 \cdots i_{k-1} 0$ is a prefix of the time period i then the secret key $sk_{\mathcal{T},i_0 i_1 \cdots i_{k-1} 1}$ is included in $SK_{\mathcal{T},i}$, this setting ensures that a user can compute his private key for the next time from the current private key.

3.2 fs-ABE: Syntax

A fs-ABE scheme is specified by five algorithms: Setup, Encryption, Key Generation, Update, Decryption:

Setup This algorithm takes as input an implicit security parameter and the total number of time periods N, and outputs the public parameters PK and the master key MK. The master key MK will be known only to the Private Key Generator (PKG).
Encryption The sender takes as input a message m, a set γ of attributes, the current time period i and the public parameters PK, and outputs a pair $\langle i, C\rangle$ as the ciphertext.
Key Generation A user requests to join the fs-ABE system at time i. The PKG takes as input an access tree $\mathcal{T}$, the master key MK and the public parameters PK, and computes an initial secret key $SK_{\mathcal{T},0}$. If the joining time $i \neq 0$, then the PKG evolves the initial secret key $SK_{\mathcal{T},0}$ to obtain the secret key $SK_{\mathcal{T},i}$ for time i and outputs $SK_{\mathcal{T},i}$, else the PKG directly outputs the initial secret key $SK_{\mathcal{T},0}$.
Update At the end of time period i, the receiver uses the secret key $SK_{\mathcal{T},i}$ to compute his secret key $SK_{\mathcal{T},i+1}$ for the next time period $i+1$, then erases the current secret key $SK_{\mathcal{T},i}$.
Decryption The receiver takes as input the ciphertext $\langle i, C\rangle$, the secret key $SK_{\mathcal{T},i}$ and the public parameters PK, and outputs the message m if $\mathcal{T}(\gamma) = 1$.

The standard correctness condition must be satisfied, namely for any (PK,MK) generated by Setup, any $SK_{\mathcal{T},0}$ output by Key Generation, any secret key $SK_{\mathcal{T},i}$ generated by Update for the time period i and any message m, we have

$$\text{Decryption}(\langle i, C\rangle, SK_{\mathcal{T},i}, \text{PK}) = m, \text{ where } \langle i, C\rangle \leftarrow \text{Encryption}(m, \gamma, i, \text{PK}).$$

3.3 fs-ABE: Security Model

We use a game between a challenger and an adversary to model the security of fs-ABE scheme. We say a fs-ABE scheme is semantically secure against

adversaries who adaptively choose the ciphertext, attributes, and time period, if all polynomial time adversaries have at most a negligible advantage against the challenger in the following game.

Setup The challenger runs the Setup algorithm of fs-ABE and gives the public parameters PK to the adversary. It keeps the master key to itself.
Phase 1 The adversary is allowed to issue the following two types of queries:

1. Private key query $SK_{\mathcal{T},j}$ associated with an access tree $\mathcal{T}$ and a time period j: the challenger runs the Key Generation algorithm to generate the private key $SK_{\mathcal{T},0}$ corresponding to the tree $\mathcal{T}$ and the initial time period 0, then recursively runs the Update algorithm to generate the private key $SK_{\mathcal{T},j}$, and sends $SK_{\mathcal{T},j}$ to the adversary.

2. Decryption query $(\mathcal{T}, \langle j, C\rangle)$: the challenger runs the Key Generation algorithm to generate the private key $SK_{\mathcal{T},0}$ corresponding to the tree $\mathcal{T}$ and the initial time period 0, recursively runs the Update algorithm to generate the private key $SK_{\mathcal{T},j}$, then runs the Decryption algorithm to decrypt $\langle j, C\rangle$ using $SK_{\mathcal{T},j}$, and sends the result to the adversary.

These queries may be issued adaptively. The adversary is allowed to query for any access tree and any time period.
Challenge Once the adversary decides that Phase 1 is over, it submits two equal length messages m_0, m_1, a set γ of attributes and a time period i on which it wishes to be challenged. The constraint is that no private key query has been issued for the access tree $\mathcal{T}$ such that $\mathcal{T}(\gamma) = 1$ for any time $j \leq i$.

The challenger flips a random coin $b \in \{0, 1\}$, and sets

$$\langle i, C^*\rangle = \text{Encryption}(m_b, \gamma, i, \text{PK}).$$

It sends $\langle i, C^*\rangle$ as a challenge to the adversary.
Phase 2 The adversary issues more queries:

1. Private key query $SK_{\mathcal{T},j}$, where the access tree $\mathcal{T}$ and the time period j are under the same restriction as in Challenge: the challenger responds as in Phase 1.

2. Decryption query $(\mathcal{T}, \langle j, C\rangle) \neq (\mathcal{T}^*, \langle i, C^*\rangle)$, where $\mathcal{T}^*$ is the access tree such that $\mathcal{T}^*(\gamma) = 1$: the challenger responds as in Phase 1.

Guess The adversary outputs a guess b' of b.

The adversary wins the game if $b = b'$. We define its advantage in this game to be $|\Pr[b = b'] - \frac{1}{2}|$. A weaker type of security model against selective chosen plaintext, a set of attributes and time period attack differs from the above model, in which the adversary declares the set of attributes and the time period on that it wishes to be challenged before the Setup in the game, and the adversary is not allowed to issue decryption queries in Phase 1 and Phase 2.

For security proof, we reduce the security of fs-ABE to the security of GPSW-ABE [10]. The security model for GPSW-ABE is similar to the weaker type of model for fs-ABE except that there is no time involved in the former one.

4 Some Trivial Forward-Secure Schemes

In this section, we take a quick look at three trivial forward-secure ABE constructions, and explain why these schemes are not good.

Scheme I. Consider a scheme based on any KP-ABE or CP-ABE schme (In fact, this construction can work for any other cryptosystem). A user is given a different private key per time period. This scheme is forward-secure. However, it has the following issues. First, a user cannot update the private key autonomously, and the workload of the PKG is increased. Second, the key size of the private key grows linearly with the number N of time periods, it is not good.

Scheme II. Consider a scheme based on GPSW-ABE [10]. During a time period i (where $0 \leq i \leq N-1$), the access tree in the private key of a user is original-tree $\mathcal{T}$ AND (time period i OR time period $i+1$ OR $\cdots$ OR time period $N-1$). In this scheme, a user can update the private key for the next time period $i+1$ from the current private key. This scheme is also forward-secure. However, the key size of the private key grows linearly with N, that is the same as Scheme I.

Scheme III. Our final trivial scheme is still based on GPSW-ABE. The initial private key of a user is composed of N secret keys, which are associated with N access trees: $\mathcal{T}$ AND time period 0, $\mathcal{T}$ AND time period 1, $\cdots$, $\mathcal{T}$ AND time period $N-1$. At the end of every time period, the corresponding secret key is erased. During a time period i, the remaining secret keys are all used to decrypt a ciphertext. Thus, the access tree in the private key for the time period i is equivalent to the tree ($\mathcal{T}$ AND time period i) OR ($\mathcal{T}$ AND time period $i+1$) OR $\cdots$ OR ($\mathcal{T}$ AND time period $N-1$), that is, $\mathcal{T}$ AND (time period i OR time period $i+1$ OR $\cdots$ OR time period $N-1$).

We see this scheme is equivalent to Scheme II, and it is not good due to the key size.

All the above trivial schemes are not the fs-ABE we needed. For simplicity, the constructions of fs-ABE mentioned later do not include these trivial schemes. In our construction, a user can update the private key autonomously, and the key size of the private key only grows polynomially with $\log N$. In addition, the secret keys composed of the private key are not all used to decrypt. Actually, only one secret key is used to do this, and the other secret keys are used to compute the private key for the next time period.

5 Generalization on Forward-Secure Compatibility

In this section, we discuss what properties should an ABE scheme have in order to support forward security, which we refer to as *forward-secure compatibility* or FS-compatibility. The two requirements are *the ability to delegate decryption keys* and *the extensibility of attributes*, which are explained next.

Delegation of private keys. In ABE schemes, the delegation of private keys means that a user who has a private key for an access tree $\mathcal{T}$ (for KP-ABE schemes) or a set S of attributes (for CP-ABE schemes) can compute a new private key for a more restrictive access tree than $\mathcal{T}$ or a subset of S. The ability of delegating the decryption capability to others (i.e., the delegation of private keys) in ABE schemes is a must for FS-compatibility. This requirement is because in a forward-secure scheme, a user needs to update his private key at the end of each time period; he must generate a new private key for the next time period using his current private key by himself without contacting the PKG. The delegation property is used in this process. In the construction (see Section 6) of our fs-ABE, the Update algorithm calls the Compute Next algorithm to compute the secret keys $sk_{\mathcal{T},ww_{(l+1)}}$ from $sk_{\mathcal{T},w}$ using the delegation property. Notice that the new time attribute "(l+1)-$w_{(l+1)}$" is added during this process. Most of the key-policy ABE schemes based on GPSW-ABE [10] can do this.

Extensibility of attributes. However, the delegation property alone is not a sufficient condition for FS-compatibility. For example, the CP-ABE scheme in [13] has the property of delegation of private keys. However, it cannot be converted to a non-trivial forward secure one in our construction. The reason is that new time attribute cannot be added to the existing set

of attributes, which are used to describe necessary secret keys. In [13], the existing secret key associated with a set of attributes can only be used to generate a secret key for a *subset* of attributes – however, new attribute cannot be introduced. In constrast, for KP-ABE schemes the new time attribute can be added to the access tree associated with the existing secret keys – satisfying the attribute-extensibility requirement. This property allows one to construct the required secret keys for future time periods. How to support CP-ABE schemes with non-trivial forward security remains an interesting challenge.

6 A Forward-secure ABE Construction

In this section, we first give some mathematical preliminaries, then present the construction of a concrete forward-secure attribute-based encryption scheme.

6.1 Preliminaries

The security of our fs-ABE scheme is based on the decisional bilinear Diffie-Hellman (DBDH) assumption. We first introduce the concept of bilinear pairings.

Bilinear Pairings Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p. A bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a map with the following properties:

1. Bilinear: $\forall u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: The map does not send all pairs in $\mathbb{G}_1 \times \mathbb{G}_1$ to the identity in $\mathbb{G}_2$.
3. Computable: $\forall u, v \in \mathbb{G}_1$, $e(u, v)$ can be efficiently computed.

We call $\mathbb{G}_1$ a bilinear group if the group operation in $\mathbb{G}_1$ is efficiently computable.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption The DBDH assumption is that all probabilistic polynomial time algorithms have at most a negligible advantage to distinguish the tuple $(g^a, g^b, g^c, e(g, g)^{abc})$ from the tuple $(g^a, g^b, g^c, e(g, g)^z)$, where g is a generator of $\mathbb{G}_1$, $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear pairing, and a, b, c, z are chosen from $\mathbb{Z}_p$ randomly. The advantage of an algorithm $\mathcal{A}$ is defined as

$$\big|\Pr[\mathcal{A}(g^a, g^b, g^c, e(g, g)^{abc}) = 1] - \Pr[\mathcal{A}(g^a, g^b, g^c, e(g, g)^z) = 1]\big|.$$

6.2 fs-ABE: Construction

Let $\mathbb{G}_1$ be a bilinear group of prime order p with a generator g. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ denote the bilinear pairing. The size of the groups is determined by a security parameter κ. We define the Lagrange coefficient $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$ for $i \in \mathbb{Z}_p$ and a set S of elements in $\mathbb{Z}_p$. This construction uses all elements of $\mathbb{Z}_p^*$ as attributes, and it also allows us to apply a collision resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$ so that we can use arbitrary strings as attributes. The message will be encrypted under a time period i and a set γ of n elements of $\mathbb{Z}_p^*$ [1]. The construction is shown below.

Setup (n, d) Let $g_1 = g^y$, where y is chosen randomly from $\mathbb{Z}_p$. Choose random $g_2 \in \mathbb{G}_1$, and choose $t_1, t_2, \cdots, t_{n+d+1}$ uniformly at random from $\mathbb{G}_1$. We define a function

$$T(X) = g_2^{X^{n+d}} \prod_{i=1}^{n+d+1} t_i^{\Delta_{i,Q}(X)},$$

where $Q = \{1, 2, \cdots, n+d+1\}$. This algorithm outputs $g_1, g_2, t_1, t_2, \cdots, t_{n+d+1}$ as the public parameters PK and y as the master key MK. The master key will be known only to the Private Key Generator (PKG).

Encryption $(m, \gamma, i, \mathrm{PK})$ Let $\langle i \rangle = i_1 i_2 \cdots i_d$. Choose random $s \in \mathbb{Z}_p$. For a message $m \in \mathbb{G}_2$, this algorithm outputs $\langle i, C \rangle$ as the ciphertext, where

$$\begin{aligned} C \;=\; & \big(\gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}, C' = me(g_1, g_2)^s, \\ & C'' = g^s, \{C_k = T(k)^s\}_{k \in \gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}}\big). \end{aligned}$$

Key Generation $(\mathcal{T}, \mathrm{MK}, \mathrm{PK})$ The PKG computes an initial private key associated with the access tree $\mathcal{T}$ for the user, so that a message encrypted under a set γ of attributes and the time period 0 can be decrypted by the user if and only if the access tree $\mathcal{T}$ can be satisfied by the set γ of attributes.

Choose a degree $d_x = k_x - 1$ of polynomial q_x for each node x in the tree $\mathcal{T}$ in a top down manner, where k_x is the threshold value of the node x. For the root node r, we completely define the polynomial q_r by setting

[1] Strictly speaking, γ is a set of n elements of $\mathbb{Z}_p^* \backslash \{\text{“1-0”}, \text{“1-1”}, \cdots, \text{“d-0”}, \text{“d-1”}\}$, where $\{\text{“1-0”}, \text{“1-1”}, \cdots, \text{“d-0”}, \text{“d-1”}\}$ denote all possible components of the d-bit representation of a time period. These components as attributes associated with time are also elements of $\mathbb{Z}_p^*$. For clarity, we use these symbols to denote the components instead of using the numbers in $\mathbb{Z}_p^*$.

$q_r(0) = y$ and d_r other points of q_r randomly. For any other node x, we completely define q_x by setting $q_x(0) = q_{\text{parent}(x)}(\text{index}(x))$ and d_x other points randomly. We add a new $(1,1)$ threshold gate z above the root node r of the tree $\mathcal{T}$, which becomes the parent of r. Define the polynomial associated with z: $q_z(X) \equiv y$ such that $q_r(0) = q_z(1)$ satisfied to the above constraint, where 1 is the index of r as a child of z. We use $\mathcal{T}_\varepsilon$ to denote the new tree to which z is added.

After all the polynomials are decided, for each leaf node x of the tree $\mathcal{T}_\varepsilon$, let

$$\begin{aligned} D_x^{(0)} &= g_2^{q_x(0)} \cdot T(\text{att}(x))^{r_x} \\ R_x^{(0)} &= g^{r_x}, \end{aligned}$$

where r_x is chosen randomly from $\mathbb{Z}_p$. Let the secret key $sk_{\mathcal{T},\varepsilon}$ associated with the tree $\mathcal{T}_\varepsilon$ as

$$sk_{\mathcal{T},\varepsilon} = \Big(\{D_x^{(0)}, R_x^{(0)}\}_{x \text{ is a leaf node of } \mathcal{T}}, \emptyset\Big).$$

Using $(sk_{\mathcal{T},\varepsilon}, \mathcal{T}_\varepsilon, \varepsilon)$, recursively apply algorithm **Compute Next** (defined below) to obtain the private key

$$SK_{\mathcal{T},0} = \big(sk_{\mathcal{T},\langle 0\rangle}, \{sk_{\mathcal{T},1}, sk_{\mathcal{T},01}, \cdots, sk_{\mathcal{T},0^{d-1}1}\}\big),$$

which is associated with the tree $\mathcal{T}$ and the time period 0. Output $SK_{\mathcal{T},0}$, and erase all other information.

Notice that if the user joins the fs-ABE system at time $i \neq 0$, then PKG recursively runs the **Update** (defined later) algorithm using $SK_{\mathcal{T},0}$ to obtain $SK_{\mathcal{T},i}$ associated with time i, and gives $SK_{\mathcal{T},i}$ to the user. For simplicity, we assume that the user joins the system at time 0.

Compute Next $(sk_{\mathcal{T},w}, \mathcal{T}_w, w)$ This algorithm takes a prefix $w = w_0 w_1 \cdots w_l$ of some time period, where $w_0 = \varepsilon, 0 \leq l \leq d-1$, the access tree $\mathcal{T}_w$ and the secret key $sk_{\mathcal{T},w}$ associated with $\mathcal{T}_w$ as input, and outputs the secret keys associated with the trees $\mathcal{T}_{w0}$ and $\mathcal{T}_{w1}$. Recall that if $w \neq \varepsilon$ then $\mathcal{T}_w$ denotes the tree that is obtained by adding the attributes "1-w_1", "2-w_2", $\cdots$, "l-w_l" in turn to the tree $\mathcal{T}_\varepsilon$ in the Key Generation algorithm as children of the root node z of $\mathcal{T}_\varepsilon$, where "1-w_1", "2-w_2", $\cdots$, "l-w_l" are the components of $w = w_1 w_2 \cdots w_l$. Notice that the root node z of the tree $\mathcal{T}_w$ is a $(l+1, l+1)$ threshold gate. We add the attribute "(l+1)-$w_{(l+1)}$" to the tree $\mathcal{T}_w$ to convert the root node z from $(l+1, l+1)$-gate to $(l+2, l+2)$-gate, and obtain the tree $\mathcal{T}_{ww_{(l+1)}}$, where $w_{(l+1)}$ represents a bit 0 or 1.

Compared to $\mathcal{T}_w$, the trees $\mathcal{T}_{w0}, \mathcal{T}_{w1}$, add one new time attribute respectively. This method needs to compute the secret keys $sk_{\mathcal{T},w0}$ and $sk_{\mathcal{T},w1}$ for $\mathcal{T}_{w0}$ and $\mathcal{T}_{w1}$. Specifically, it needs to define the polynomial for the new time attribute. In addition, the polynomials associated with the existing attributes need to be refreshed. Then, it uses the constant terms of the polynomials associated with all the attributes in the tree to define the components of the secret key associated with the tree. Recall that in an access tree an attribute is represented by a leaf node.

Let $q_x^{(l)}(X)$ denote the polynomial associated with the node x in the tree $\mathcal{T}_w$, and let $d_x^{(l)}$ denote the degree of $q_x^{(l)}(X)$. Define $q_x^{(0)}(X) = q_x(X)$, $d_x^{(0)} = d_x$ if x is in $\mathcal{T}$or $x = z$.

1. We refresh the polynomials for all the existing nodes and define the polynomial for the new node as follows.

   (a) (refresh) Refresh the polynomial for z:

   $$q_z^{(l+1)}(X) = (-\frac{1}{l+2}X + 1)q_z^{(l)}(X) + p_z^{(l+1)}(X),$$

   where $p_z^{(l+1)}(X)$ is a random polynomial of degree $d_z^{(l+1)} = d_z^{(l)} + 1$ such that $p_z^{(l+1)}(0) = 0$. Thus, $q_z^{(l+1)}(0) = q_z^{(l)}(0) = \cdots = q_z^{(0)}(0) = y$, where y is the master key.

   (b) (refresh) For each node x in the tree $\mathcal{T}$, refresh the polynomial for x:

   $$q_x^{(l+1)}(X) = (-\frac{1}{l+2} \cdot 1 + 1)q_x^{(l)}(X) + p_x^{(l+1)}(X),$$

   where $p_x^{(l+1)}(X)$ is a random polynomial of degree $d_x^{(l+1)} = d_x$ such that $p_x^{(l+1)}(0) = p_{\text{parent}(x)}^{(l+1)}(\text{index}(x))$. These polynomials are chosen in a top-down manner.

   (c) (refresh) If $w = \varepsilon$, then go to step 1d, else for each node "k-w_k" $(1 \le k \le l)$, refresh the polynomial for "k-w_k":

   $$q_{\text{“k-}w_k\text{”}}^{(l+1)}(X) = (-\frac{1}{l+2} \cdot (k+1) + 1)q_{\text{“k-}w_k\text{”}}^{(l)}(X) + p_{\text{“k-}w_k\text{”}}^{(l+1)}(X),$$

   where $p_{\text{“k-}w_k\text{”}}^{(l+1)}(X)$ is a polynomial of degree 0 such that $p_{\text{“k-}w_k\text{”}}^{(l+1)}(0) = p_z^{(l+1)}(k+1)$.

(d) (define) For the new node "(l+1)-$w_{(l+1)}$", define the polynomial for "(l+1)-$w_{(l+1)}$":

$$q^{(l+1)}_{\text{“(l+1)-}w_{(l+1)}\text{”}}(X) = 0 + p^{(l+1)}_{\text{“(l+1)-}w_{(l+1)}\text{”}}(X),$$

where $p^{(l+1)}_{\text{“(l+1)-}w_{(l+1)}\text{”}}(X)$ is a polynomial of degree 0 such that $p^{(l+1)}_{\text{“(l+1)-}w_{(l+1)}\text{”}}(0) = p^{(l+1)}_z(l+2)$.

All the above polynomials satisfy to the constraint in the Key Generation algorithm.

2. If $w \neq \varepsilon$, then let

$$sk_{\mathcal{T},w} = \Big(\{D^{(l)}_x, R^{(l)}_x\}_{x \text{ is a leaf node of } \mathcal{T}}, \{D^{(l)}_{\text{“k-}w_k\text{”}}, R^{(l)}_{\text{“k-}w_k\text{”}}\}_{1\leq k\leq l}\Big).$$

Let the secret key associated with the tree $\mathcal{T}_{ww_{(l+1)}}$ as

$$sk_{\mathcal{T},ww_{(l+1)}} = \Big(\{D^{(l+1)}_x, R^{(l+1)}_x\}_{x \text{ is a leaf node of } \mathcal{T}}, \{D^{(l+1)}_{\text{“k-}w_k\text{”}}, R^{(l+1)}_{\text{“k-}w_k\text{”}}\}_{1\leq k\leq l+1}\Big).$$

Next, we use the constant terms of the polynomials associated with all the leaf nodes in the tree $\mathcal{T}_{ww_{(l+1)}}$ to define the components of the secret key $sk_{\mathcal{T},ww_{(l+1)}}$. We refresh the D, R values for all the existing nodes and define the D, R values for the new node as follows.

(a) (refresh) For each leaf node x in the tree $\mathcal{T}$, refresh D, R values for x:

$$\begin{aligned} D^{(l+1)}_x &= (D^{(l)}_x)^{-\frac{1}{l+2}\cdot 1+1} \cdot g_2^{p^{(l+1)}_x(0)} \cdot T(\mathrm{att}(x))^{r^{(l+1)}_x} \\ R^{(l+1)}_x &= (R^{(l)}_x)^{-\frac{1}{l+2}\cdot 1+1} \cdot g^{r^{(l+1)}_x} \end{aligned}$$

where $r^{(l+1)}_x \in \mathbb{Z}_p$ is chosen randomly.

(b) (refresh) If $w = \varepsilon$, then go to step 2c, else for each node "k-w_k" $(1 \leq k \leq l)$, refresh D, R values for "k-w_k":

$$\begin{aligned} D^{(l+1)}_{\text{“k-}w_k\text{”}} &= (D^{(l)}_{\text{“k-}w_k\text{”}})^{-\frac{1}{l+2}\cdot(k+1)+1} \cdot g_2^{p^{(l+1)}_{\text{“k-}w_k\text{”}}(0)} \cdot T(\mathrm{att}(\text{“k-}w_k\text{”}))^{r^{(l+1)}_{\text{“k-}w_k\text{”}}} \\ R^{(l+1)}_{\text{“k-}w_k\text{”}} &= (R^{(l)}_{\text{“k-}w_k\text{”}})^{-\frac{1}{l+2}\cdot(k+1)+1} \cdot g^{r^{(l+1)}_{\text{“k-}w_k\text{”}}} \end{aligned}$$

where $r^{(l+1)}_{\text{“k-}w_k\text{”}} \in \mathbb{Z}_p$ is chosen randomly.

(c) (define) For the new node "(l+1)-$w_{(l+1)}$", define D, R values for "(l+1)-$w_{(l+1)}$":

$$D^{(l+1)}_{\text{“(l+1)-}w_{(l+1)}\text{”}} = g_2^{p^{(l+1)}_{\text{“(l+1)-}w_{(l+1)}\text{”}}(0)} \cdot T(\text{att}(\text{“(l+1)-}w_{(l+1)}\text{”}))^{r^{(l+1)}_{\text{“(l+1)-}w_{(l+1)}\text{”}}}$$

$$R^{(l+1)}_{\text{“(l+1)-}w_{(l+1)}\text{”}} = g^{r^{(l+1)}_{\text{“(l+1)-}w_{(l+1)}\text{”}}}$$

where $r^{(l+1)}_{\text{“(l+1)-}w_{(l+1)}\text{”}} \in \mathbb{Z}_p$ is chosen randomly.

All the above D, R values are in the same form as in the Key Generation algorithm.

The algorithm outputs the secret keys $sk_{\mathcal{T},w0}, sk_{\mathcal{T},w1}$ associated with the trees $\mathcal{T}_{w0}, \mathcal{T}_{w1}$.

Update $(SK_{\mathcal{T},i}, i+1)$(where $i < N-1$) Let $\langle i \rangle = i_0 i_1 \cdots i_d$, where $i_0 = \varepsilon$. Let $SK_{\mathcal{T},i} = \big(sk_{\mathcal{T},\langle i \rangle}, \{sk_{\mathcal{T},i_0 i_1 \cdots i_{k-1}1}\}_{i_k=0}\big)$. Erase $sk_{\mathcal{T},\langle i \rangle}$. We distinguish two cases. If $i_d = 0$, simply output the remaining keys as the key $SK_{\mathcal{T},i+1}$ for the next period. Otherwise, let $\tilde{k}$ be the largest value such that $i_{\tilde{k}} = 0$ (such $\tilde{k}$ must exist since $i < N-1$). Let $i' = i_0 i_1 \cdots i_{\tilde{k}-1}1$. Notice that $sk_{\mathcal{T},i'}$ is included as part of $SK_{\mathcal{T},i}$. Using $(sk_{\mathcal{T},i'}, \mathcal{T}_{i'}, i')$, recursively apply algorithm **Compute Next** to generate keys $sk_{\mathcal{T},i'1}, sk_{\mathcal{T},i'01}, \cdots, sk_{\mathcal{T},i'0^{d-\tilde{k}-1}1}, sk_{\mathcal{T},i'0^{d-\tilde{k}}}$. Erase $sk_{\mathcal{T},i'}$ and output the remaining keys as $SK_{\mathcal{T},i+1}$.

Decryption $(\langle i, C \rangle, SK_{\mathcal{T},i})$ This algorithm outputs m if and only if $\mathcal{T}(\gamma) = 1$. Let $\langle i \rangle = i_0 i_1 \cdots i_d$, where $i_0 = \varepsilon$. Let

$$SK_{\mathcal{T},i} = \big(sk_{\mathcal{T},\langle i \rangle}, \{sk_{\mathcal{T},i_0 i_1 \cdots i_{k-1}1}\}_{i_k=0}\big),$$

$$sk_{\mathcal{T},\langle i \rangle} = \Big(\{D^{(d)}_x, R^{(d)}_x\}_{x \text{ is a leaf node of } \mathcal{T}}, \{D^{(d)}_{\text{“k-}w_k\text{”}}, R^{(d)}_{\text{“k-}w_k\text{”}}\}_{1 \le k \le d}\Big).$$

We first define a recursive algorithm DecryptNode$(C, sk_{\mathcal{T},\langle i \rangle}, x)$, where $C = \big(\gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}, C', C'', \{C_k\}_{k \in \gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}}\big)$, and x is a node in the tree $\mathcal{T}_{\langle i \rangle}$.

- If x is a leaf node, then

$$\text{DecryptNode}(C, sk_{\mathcal{T},\langle i \rangle}, x) = \begin{cases} \frac{e(D^{(d)}_x, C'')}{e(R^{(d)}_x, C_k)} & \text{if } k = \text{att}(x) \in \gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\} \\ \perp & \text{otherwise} \end{cases}$$

- If x is an interior node, then DecryptNode$(C, sk_{\mathcal{T},\langle i\rangle}, x)$ is defined as follows: If x is not satisfied, then DecryptNode$(C, sk_{\mathcal{T},\langle i\rangle}, x)$ returns $\perp$. Otherwise, call DecryptNode$(C, sk_{\mathcal{T},\langle i\rangle}, x')$ for all children x' of x and denote the output by $F_{x'}$. Let S_x denote a set that consists of k_x nodes x' such that $F_{x'} \neq \perp$. We define

$$\text{DecryptNode}(C, sk_{\mathcal{T},\langle i\rangle}, x) = \prod_{x' \in S_x} F_{x'}^{\Delta_{j,S'_x}(0)}$$

where $j = \text{index}(x')$ and $S'_x = \{\text{index}(x') : x' \in S_x\}$.

The decryption algorithm calls the DecryptNode with $(C, sk_{\mathcal{T},\langle i\rangle}, z)$, where z is the root of the tree $\mathcal{T}_{\langle i\rangle}$. If the output of the DecryptNode is not $\perp$, then we use it to divide into C' and output the result. The correctness is shown in the appendix.

Proof of Security We reduce the security of fs-ABE to the security of GPSW-ABE [10]. We prove that the fs-ABE scheme is secure in the selective chosen plaintext, a set of attributes and time period model. In section 8, we will describe that how a selective chosen plaintext secure fs-ABE scheme can be transformed into one that is secure against adaptive chosen ciphertext adversaries.

The security is proven in a game model. The adversary $\mathcal{A}$ who can break the fs-ABE scheme can be used to construct an adversary $\mathcal{B}$ who attacks the GPSW-ABE scheme with a non-negligible advantage. $\mathcal{B}$ is given the public parameters of the GPSW-ABE scheme by its challenger, then it simulates the fs-ABE environment for $\mathcal{A}$. Although $\mathcal{B}$ does not have the master key of fs-ABE, $\mathcal{B}$ needs to answer the private key queries and the challenge ciphertext query from $\mathcal{A}$. $\mathcal{B}$'s responses have to be well-formed, i.e., compliant with the specifications of the fs-ABE scheme.

Theorem 1 *If an adversary has advantage ϵ to attack the fs-ABE scheme in the selective chosen plaintext, a set of attributes and time period model, then a simulator can be constructed to attack the GPSW-ABE scheme in the selective chosen plaintext, a set of attributes model with the same advantage ϵ.*

The full proof is shown in the appendix. We summarize our proof strategy here. The simulator $\mathcal{B}$ needs to answer the private key queries $SK_{\mathcal{T},j}$ in two cases: 1. $\mathcal{T}(\gamma) = 1$ and $j > i$. 2. $\mathcal{T}(\gamma) = 0$. To make $\mathcal{B}$'s responses well-formed, we define a procedure PolyUnsat, which sets up a polynomial for each node of an unsatisfied access tree. For case 1, $\mathcal{B}$ runs PolyUnsat on the

trees $\mathcal{T}_{\langle j \rangle}$ and $\{\mathcal{T}_{j_0 j_1 \cdots j_{k-1} 1}\}_{j_k=0}$, then constructs the secret keys associated with these trees using the polynomials defined by PolyUnsat. $\mathcal{B}$ passes the set of above secret keys to $\mathcal{A}$ as $SK_{\mathcal{T},j}$. For case 2, $\mathcal{B}$ runs PolyUnsat on the tree $\mathcal{T}$, and constructs the secret key associated with $\mathcal{T}$, then uses this key to obtain $SK'_{\mathcal{T},0}$ as in Key Generation of fs-ABE. Next, $\mathcal{B}$ runs Update in fs-ABE with $SK'_{\mathcal{T},0}$ to generete $SK'_{\mathcal{T},j}$, and passes $SK'_{\mathcal{T},j}$ to $\mathcal{A}$ as $SK_{\mathcal{T},j}$. The private key $SK_{\mathcal{T},j}$ in two cases both have identical distribution to that of the fs-ABE scheme. We also manage to make the simulator $\mathcal{B}$'s generation of the public parameters and the challenge ciphertext identical to that of the fs-ABE scheme. Thus, the Theorem 1 holds.

7 Individualized Forward-secure Attribute-Based Encryption (ifs-ABE)

In fs-ABE described above, all attributes need to be updated according to the same schedule. We further extend our fs-ABE scheme to support more flexible and individualized key-update schedules, specifically each attribute may enjoy its own forward-secure key-updating precision. Our main motivation is to achieve more efficient operations. We refer to this new scheme as individualized fs-ABE (ifs-ABE). The ifs-ABE scheme provides the same security as fs-ABE in the fs-ABE security model. The security of the ifs-ABE scheme can be proven using a similar method as in Theorem 1. We outline our scheme next.

In an ifs-ABE scheme, each attribute has its own time precision. If the standard time period is i, then for an attribute x with the time precision d_x ($d_x \leq d$) the local time period is $i|_{d_x}$, where $i|_{d_x}$ is the integer converted from the d_x-bit prefix $i_1 i_2 \cdots i_{d_x}$ of $\langle i \rangle = i_1 i_2 \cdots i_d$. In such a scheme, the private key of a user is evolved with time. At the standard time period 0 a user is issued an initial private key associated with an access tree by the Private Key Generator (PKG). At the end of each standard time period he updates his private key for the next standard time period and erases the current key. For an attribute x with the time precision d_x, the corresponding component of the private key is not always updated when the standard time period changes. Specifically, if the next standard time period is multiple of 2^{d-d_x} then it will be updated. As an example, let the standard time precision $d = 3$ and the time precision d_x associated with an attribute x be 2. When the standard time period changes from $\langle 4 \rangle = 100$ to $\langle 5 \rangle = 101$, the component corresponding to x in the private key will not be updated because for the attribute x the local time period does not change. When the

standard time period changes from $\langle 5 \rangle = 101$ to $\langle 6 \rangle = 110$, the component will be updated because the local time period changes from $\langle 5 \rangle|_2 = 10$ to $\langle 6 \rangle|_2 = 11$, i.e., from 2 to 3.

We provide the detailed description on such an ifs-ABE scheme in the following. Let $\mathbb{G}_1$ be a bilinear group of prime order p with a generator g. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ denote the bilinear pairing. The size of the groups is determined by a security parameter κ. We define the Lagrange coefficient $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$ for $i \in \mathbb{Z}_p$ and a set S of elements in $\mathbb{Z}_p$ as before. This construction uses elements of $\mathbb{Z}_p^*$ as attributes. The message will be encrypted under a standard time period i and a set γ of n elements of $\mathbb{Z}_p^*$. The construction is shown below.

Setup (n, d) This algorithm runs the Setup of fs-ABE with (n, nd). Let (PK,MK) denote the output of that process. Output PK as the public parameters and MK as the master key. The master key MK will be known only to the Private Key Generator (PKG).

Encryption $(m, \gamma, i, \mathrm{PK})$ Let $\gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_n\}$ and $\langle i \rangle = i_1 i_2 \cdots i_d$. For a message $m \in \mathbb{G}_2$, this algorithm chooses random $s \in \mathbb{Z}_p$ and outputs $\langle i, C \rangle$ as the ciphertext, where

$$\begin{aligned} C \;=\; & \big(\gamma \cup \{\text{“}\gamma_1\text{-1-}i_1\text{”}, \cdots, \text{“}\gamma_1\text{-d}_{\gamma_1}\text{-}i_{d_{\gamma_1}}\text{”}, \cdots, \text{“}\gamma_n\text{-1-}i_1\text{”}, \cdots, \text{“}\gamma_n\text{-d}_{\gamma_n}\text{-}i_{d_{\gamma_n}}\text{”}\}, \\ & C' = me(g_1, g_2)^s, C'' = g^s, \\ & \{C_k = T(k)^s\}_{k \in \gamma \cup \{\text{“}\gamma_1\text{-1-}i_1\text{”}, \cdots, \text{“}\gamma_1\text{-d}_{\gamma_1}\text{-}i_{d_{\gamma_1}}\text{”}, \cdots, \text{“}\gamma_n\text{-1-}i_1\text{”}, \cdots, \text{“}\gamma_n\text{-d}_{\gamma_n}\text{-}i_{d_{\gamma_n}}\text{”}\}}\big). \end{aligned}$$

Notice that “γ_j-1-i_1”, $\cdots$, “γ_j-d_{γ_j}-$i_{d_{\gamma_j}}$” for $j = 1$ to n denote the components of $\langle i \rangle|_{d_{\gamma_j}} = i_1 i_2 \cdots i_{d_{\gamma_j}}$ corresponding to the attribute γ_j.

Key Generation $(\mathcal{T}, \mathrm{MK}, \mathrm{PK})$ The PKG computes an initial private key associated with the access tree $\mathcal{T}$ for the user, so that a message encrypted under a set γ of attributes and the standard time period 0 can be decrypted by the user if and only if the access tree $\mathcal{T}$ can be satisfied by the set γ of attributes.

Choose a polynomial for each node in the tree $\mathcal{T}$ as in the Key Generation of fs-ABE. For each leaf node x, add a new $(1, 1)$ threshold gate z above x, which becomes the parent of x replacing the former one y. Define the polynomial associated with z: $q_z(X) \equiv q_y(\mathrm{index}(z))$ such that $q_x(0) = q_z(1)$ satisfied to the constraint in the Key Generation of fs-ABE, where 1 is the index of x as a child of z. There is no need to modify the polynomials

associated with the ancestor nodes of y. After each leaf node is processed above, we still use $\mathcal{T}$ to denote the new tree.

For each leaf node x of the tree $\mathcal{T}$, let

$$\begin{aligned} D_x^{(0)} &= g_2^{q_x(0)} \cdot T(\mathrm{att}(x))^{r_x} \\ R_x^{(0)} &= g^{r_x}, \end{aligned}$$

where r_x is chosen randomly from $\mathbb{Z}_p$. Let $(\mathcal{T}|_x)_\varepsilon$ denote the the subtree $\mathcal{T}|_z$ of the tree $\mathcal{T}$, where z is the (1,1) threshold gate added above x. Let the secret key $sk_{\mathcal{T},x,\varepsilon}$ associated with the tree $(\mathcal{T}|_x)_\varepsilon$ as

$$sk_{\mathcal{T},x,\varepsilon} = \Big(\{D_x^{(0)}, R_x^{(0)}\}, \emptyset\Big).$$

Using $(sk_{\mathcal{T},x,\varepsilon}, (\mathcal{T}|_x)_\varepsilon, \varepsilon)$, recursively apply algorithm **Compute Next** (defined below) to obtain the private key

$$SK_{\mathcal{T},0} = \{SK_{\mathcal{T},x,0|_{d_x}}\}_{x \text{ is a leaf node of } \mathcal{T}},$$

$$SK_{\mathcal{T},x,0|_{d_x}} = \big(sk_{\mathcal{T},x,\langle 0\rangle|_{d_x}}, \{sk_{\mathcal{T},x,1}, sk_{\mathcal{T},x,01}, \cdots, sk_{\mathcal{T},x,0^{d_x-1}1}\}\big),$$

which is associated with the tree $\mathcal{T}$ and the standard time period 0. Output $SK_{\mathcal{T},0}$ and erase all other information.

Compute Next $(sk_{\mathcal{T},x,w}, (\mathcal{T}|_x)_w, w)$ This algorithm takes a prefix $w = w_0 w_1 \cdots w_l$ of some local time period, where $w_0 = \varepsilon, 0 \le l \le d_x - 1$, the access tree $(\mathcal{T}|_x)_w$ and the secret key $sk_{\mathcal{T},x,w}$ associated with $(\mathcal{T}|_x)_w$ as input, and outputs the secret keys associated with the trees $(\mathcal{T}|_x)_{w0}$ and $(\mathcal{T}|_x)_{w1}$. Notice that if $w \neq \varepsilon$ then $(\mathcal{T}|_x)_w$ denote the tree that is obtained by adding the attributes "att(x)-1-w_1", "att(x)-2-w_2", $\cdots$, "att(x)-l-w_l" in turn to the tree $(\mathcal{T}|_x)_\varepsilon$ in the Key Generation algorithm as children of the root node z of $(\mathcal{T}|_x)_\varepsilon$, where x is a leaf node of the tree $\mathcal{T}$, and "att(x)-1-w_1", "att(x)-2-w_2", $\cdots$, "att(x)-l-w_l" are the components of $w = w_1 w_2 \cdots w_l$ corresponding to x. The tree $(\mathcal{T}|_x)_w$ is shown in Figure 3.

If $w \neq \varepsilon$, then let

$$sk_{\mathcal{T},x,w} = \Big(\{D_x^{(l)}, R_x^{(l)}\}, \{D^{(l)}_{\text{“att}(x)\text{-k-}w_k\text{”}}, R^{(l)}_{\text{“att}(x)\text{-k-}w_k\text{”}}\}_{1\le k\le l}\Big).$$

Run the Compute Next of fs-ABE with $(sk_{\mathcal{T},x,w}, (\mathcal{T}|_x)_w, w)$ and obtain the secret keys $sk_{\mathcal{T}|_x,w0}$ and $sk_{\mathcal{T}|_x,w1}$ associated with the trees $(\mathcal{T}|_x)_{w0}$ and $(\mathcal{T}|_x)_{w1}$ respectively. This algorithm outputs $sk_{\mathcal{T}|_x,w0}$ and $sk_{\mathcal{T}|_x,w1}$ as the secret keys $sk_{\mathcal{T},x,w0}$ and $sk_{\mathcal{T},x,w1}$.

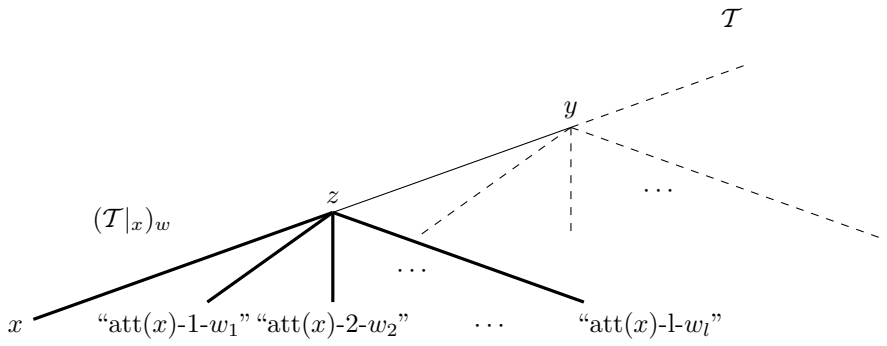


Figure 3: In the Compute-Next operation of ifs-ABE, the tree $(\mathcal{T}|_x)_w$, where x is a leaf node of $\mathcal{T}$, z is a node added above x to become the parent of x replacing the former one y in the Key Generation algorithm, and "att(x)-1-w_1", "att(x)-2-w_2", $\cdots$, "att(x)-l-w_l" are the components of $w = w_1 w_2 \cdots w_l$ corresponding to x.

Update $(SK_{\mathcal{T},i}, i+1)$(where $i < N-1$) Let $\langle i \rangle = i_0 i_1 \cdots i_d$, where $i_0 = \varepsilon$. Let

$$SK_{\mathcal{T},i} = \{SK_{\mathcal{T},x,i|_{d_x}}\}_{x \text{ is a leaf node of } \mathcal{T}},$$

$$SK_{\mathcal{T},x,i|_{d_x}} = (sk_{\mathcal{T},x,\langle i \rangle|_{d_x}}, \{sk_{\mathcal{T},x,i_0 i_1 \cdots i_{k-1}1}\}_{i_k=0})$$

where $\langle i \rangle|_{d_x} = i_0 i_1 \cdots i_{d_x}$, $i|_{d_x}$ is the integer converted from the binary string $\langle i \rangle|_{d_x}$, and $k \leq d_x$. For each leaf node x of $\mathcal{T}$, if $i \equiv 1 \mod 2^{d-d_x}$ and $0 < i < 2^d - 1$ then run the Update of fs-ABE with $(SK_{\mathcal{T}|_x,i|_{d_x}}, i|_{d_x} + 1)$ and obtain the output $SK_{\mathcal{T}|_x,i|_{d_x}+1}$ of that process as $SK_{\mathcal{T},x,i|_{d_x}+1}$. This algorithm outputs

$$\{SK_{\mathcal{T},x,i|_{d_x}+1}\}_{x \text{ is a leaf node of } \mathcal{T}}$$

as the secret key $SK_{\mathcal{T},i+1}$ associated with the tree $\mathcal{T}$ and the standard time period $i+1$.

Decryption $(\langle i, C \rangle, SK_{\mathcal{T},i})$ This algorithm outputs the message m if and only if the access tree $\mathcal{T}$ can be satisfied by the set γ of attributes in the ciphertext. Let $\langle i \rangle = i_0 i_1 \cdots i_d$, where $i_0 = \varepsilon$. Let

$$SK_{\mathcal{T},i} = \{SK_{\mathcal{T},x,i|_{d_x}}\}_{x \text{ is a leaf node of } \mathcal{T}},$$

$$SK_{\mathcal{T},x,i|_{d_x}} = \big(sk_{\mathcal{T},x,\langle i \rangle|_{d_x}}, \{sk_{\mathcal{T},x,i_0 i_1 \cdots i_{k-1}1}\}_{i_k=0}\big),$$

where $\langle i \rangle|_{d_x} = i_0 i_1 \cdots i_{d_x}$, $i|_{d_x}$ is the integer converted from the binary string $\langle i \rangle|_{d_x}$, and $k \leq d_x$. This algorithm takes

$$\{sk_{\mathcal{T},x,\langle i \rangle|_{d_x}}\}_{x \text{ is a leaf node of } \mathcal{T}}$$

as the secret key $sk_{\mathcal{T},\langle i \rangle}$ associated with the tree $\mathcal{T}_{\langle i \rangle}$ that is obtained by adding the attributes "att(x)-1-i_1", "att(x)-2-i_2", $\cdots$, "att(x)-$\mathrm{d_x}$-i_{d_x}" to the tree $\mathcal{T}$ in the Key Generation algorithm for each leaf node x of the $\mathcal{T}$. Call the function DecryptNode defined in the Decryption of fs-ABE with $(C, sk_{\mathcal{T},\langle i \rangle}, r)$, where r is the root of the tree $\mathcal{T}_{\langle i \rangle}$. If the output of the DecryptNode is not $\perp$, then we use it to divide into C' (recall that C' is a component of C) and obtain the message m.

We can verify that decryption of ifs-ABE is performed correctly as in Appendix A. The process of verification is omited. The ifs-ABE scheme can be proven secure in the security model defined in section 3.3 using a similar method to that is used to proof the security of fs-ABE. We have the following theorem.

Theorem 2 *If an adversary can break the ifs-ABE scheme in the selective chosen plaintext, a set of attributes and time period model, then a simulator can be constructed to solve the DBDH problem with a non-negligible advantage.*

8 Discussion

Convert Selective CPA Secure fs-ABE to Adaptive CCA Secure One. We describe that how to convert a fs-ABE scheme that is secure against selective chosen plaintext attack (CPA) to one that is secure against adaptive chosen ciphertext attack (CCA).

A strongly unforgeable one-time signature scheme will be used. We denote the verification key and signing key of such a signature scheme by vk and sk respectively. If a sender wants to use a set γ of attributes, the current time period i and the public parameters PK to encrypt a message, then he now uses $\gamma \cup \{\text{"vk"}\}$, i and PK to encrypt it, where "vk" is an attribute corresponding to the verification key vk. Next, the sender signs the ciphertext $\langle i, C \rangle$ using sk and takes $\big(vk, \langle i, C \rangle, \sigma\big)$ as the new ciphertext, where σ is the signature. A receiver with the private key associated with an access tree $\mathcal{T}$ and the current time period i first uses the verification key vk in the ciphertext to check if the signature is valid. If not, he rejects the ciphertext. Otherwise, the receiver adds the attribute "vk" to the tree $\mathcal{T}_{\langle i \rangle}$

as a child of the root z of $\mathcal{T}_{\langle i \rangle}$. Notice that z is converted from $(d+1, d+1)$-gate to $(d+2, d+2)$-gate. Then he computes the secret key associated with the new tree using the same method in the Compute Next algorithm, and uses it to decrypt $\langle i, C \rangle$. The receiver obtains the message if and only if the access tree $\mathcal{T}$ is satisfied by the set γ of attributes.

According to the work by Canetti et al. [30], a selective CPA secure fs-ABE scheme can be transformed into an adaptive CCA secure one by the above construction.

Private Key With Expiration Time. In the fs-ABE scheme we can add an expiration time to the private key of a user, so that during the time period time i the user can decrypt the ciphertext if and only if, i is less than the expiration time, and the access tree $\mathcal{T}$ in the private key is satisfied by the set γ of attributes in the ciphertext. As an example, let the total number of time periods $N = 2^d = 2^4$, the current time period be $\langle 12 \rangle = 1100$ and the expiration time be $\langle 13 \rangle = 1101$. As in [13] we construct a subtree corresponding to the expiration time $\langle 13 \rangle = 1101$ as shown in Figure 4, and add it to the tree $\mathcal{T}$ as a child of the root r of $\mathcal{T}$. The node r is converted from (k_r, n_r)-gate to $(k_r + 1, n_r + 1)$-gate. Notice that there is a relationship between the construction of the subtree and the bit representation of the expiration time. The PKG generates the initial private key for the user according to this new tree instead of the tree $\mathcal{T}$. The user updates his private key by himself when the time period changes. During the current time period $\langle 12 \rangle = 1100$ the subtree is satisfied by the components "1-1", "2-1", "3-0", "4-0" of $\langle 12 \rangle = 1100$, and the private key does not expire. The user can decrypt the ciphertext if and only if the access tree $\mathcal{T}$ is satisfied by the set γ of attributes in the ciphertext. If the time period is $\langle 14 \rangle = 1110$, then the subtree is not satisfied by the components "1-1", "2-1", "3-1", "4-0" of $\langle 14 \rangle = 1110$, and the private key expires. The user can not decrypt the ciphertext even if the access tree $\mathcal{T}$ is satisfied by the set γ of attributes.

Analysis of Complexities. We present the running time complexities and key sizes of fs-ABE and ifs-ABE schemes in Table 1. For both fs-ABE and ifs-ABE schemes, Key Generation and Update algorithms need to call Compute Next algorithm of fs-ABE. The time complexity of Compute Next depends on the number of all nodes in the input access tree. We show the key-generation time and key-update time in Table 1. Notice that in the ifs-ABE scheme we assume that each attribute has time precision d for the worst. If the number of all nodes in the access tree $\mathcal{T}$ of a user is large, and many attributes have time precision less than d, then the Key Generation and Update algorithms of ifs-ABE are more efficient than those of fs-ABE.

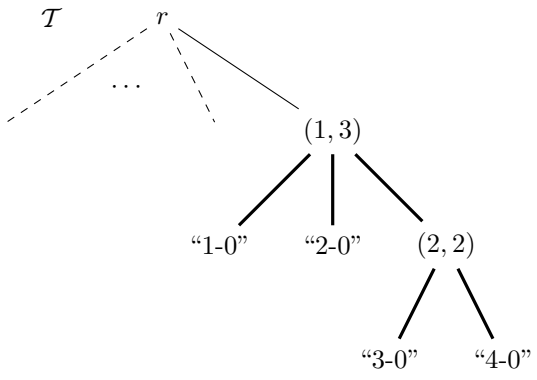


Figure 4: A subtree corresponding to the expiration time $\langle 13 \rangle = 1101$ is added to the access tree $\mathcal{T}$ as a child of the root r of $\mathcal{T}$, where (1,3), (2,2) are threshold gates.

As in GPSW-ABE [10], the decryption time of fs-ABE and ifs-ABE can be improved by the similar method. We omit the improvement procedure and show the optimized time in Table 1. The Key Update algorithm has the same computation cost $\mathcal{O}((h+\log N)\log N)$ as the Key Generation algorithm, we show that in Table 1. The Key Update algorithm cannot be replaced by the Key Generation algorithm. A user needs to update his private key on his own, without contacting the PKG.

Delegation. Our fs-ABE scheme has the property of delegation of private keys. For a private key $SK_{\mathcal{T},i}$ associated with an access tree $\mathcal{T}$ and a time period i, we can make use of the same operations in [10] to convert every secret key in $SK_{\mathcal{T},i}$ to a secret key for an access tree $\mathcal{T}'$ which is more restrictive than $\mathcal{T}$. The set of these new secret keys is just the private key $SK_{\mathcal{T}',i}$ for the access tree $\mathcal{T}'$ and the time period i.

9 Conclusions and an Open Problem

In this paper, we provided in-depth and formal descriptions on how to define, construct, and analyze a forward-secure attribute-based encryption scheme, which is a public-key encryption scheme with evolving decryption keys. The forward security protects the confidentiality of past communications against stolen decryption keys. This property is important for the attribute-based encryption paradigm. We gave several extensions to our fs-ABE scheme to further improve its usability and security.

Table 1: Time Complexities and Key Sizes

Parameters	fs-ABE	ifs-ABE
Key generation time	$\mathcal{O}((h + \log N) \log N)$	$\mathcal{O}(m \log^2 N)$
Encryption time	$\mathcal{O}(n + \log N)$	$\mathcal{O}(n \log N)$
Decryption time	$\mathcal{O}(m + \log N)$	$\mathcal{O}(m \log N)$
Key update time	$\mathcal{O}((h + \log N) \log N)$	$\mathcal{O}(m \log^2 N)$
Ciphertext length	$\mathcal{O}(n + \log N)$	$\mathcal{O}(n \log N)$
Public key size	$\mathcal{O}(n + \log N)$	$\mathcal{O}(n \log N)$
Secret key size	$\mathcal{O}((m + \log N) \log N)$	$\mathcal{O}(m \log^2 N)$

Note 1: The running time complexities and key sizes of fs-ABE and ifs-ABE, where N is the total number of time periods, n is the maximum size of the set of attributes a sender can encrypt under, m and h are respectively the number of leaf nodes and all nodes in the access tree $\mathcal{T}$ of a user.
Note 2: In the ifs-ABE scheme we assume that each attribute has time precision d for the worst, where $d = \log N$. If the number of all nodes in the access tree $\mathcal{T}$ of a user is large, and many attributes have time precision less than d, then the Key Generation and Update algorithms of ifs-ABE are more efficient than those of fs-ABE.

Not all the existing ABE schemes may be converted into a non-trivial forward secure one. We generalize the properties that an ABE scheme should possess in order to support forward security. This analysis on forward-secure compatibility of ABE schemes is general and useful beyond the concrete cryptographic construction that we presented. We pointed it out an open problem on how to support non-trivial forward secrecy in the existing ciphertext-policy ABE scheme.

An open problem. Both ABE and IBE schemes have the inherent key-escrow property – the key generator can derive the description keys for everyone in the system – thus creating a single point of failure. A compromised master key would compromise all communications. In pairing-based IBE constructions, the master key can enjoy forward security [5], which significantly mitigates the key-escrow problem. However, in our construction the master key (y) cannot evolve with time, and it is kept the same. It remains an interesting open problem on how to construct a fs-ABE scheme that preserves the confidentiality of past communications of all users even in the face of a compromised master secret. In such a scheme, the current master key cannot be used to derive decryption keys of previous time periods. This attack is not included in our current security model.

A Correctness of Decryption Algorithm

We verify that decryption of fs-ABE is performed correctly. Recall

$$\begin{aligned} C &= \big(\gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}, C' = me(g_1, g_2)^s, \\ &\quad C'' = g^s, \{C_k = T(k)^s\}_{k \in \gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}}\big). \end{aligned}$$

When decrypting, for each leaf node x, if $k = \mathrm{att}(x) \in \gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}$ then

$$\begin{aligned} \mathrm{DecryptNode}(C, sk_{\mathcal{T},\langle i\rangle}, x) &= \frac{e(D_x^{(d)}, C'')}{e(R_x^{(d)}, C_k)} \\ &= \frac{e(g_2^{q_x^{(d)}(0)} \cdot T(k)^{\bar{r}_x^{(d)}}, g^s)}{e(g^{\bar{r}_x^{(d)}}, T(k)^s)} \\ &= \frac{e(g_2^{q_x^{(d)}(0)}, g^s) \cdot e(T(k)^{\bar{r}_x^{(d)}}, g^s)}{e(g^{\bar{r}_x^{(d)}}, T(k)^s)} \\ &= e(g, g_2)^{s \cdot q_x^{(d)}(0)}, \end{aligned}$$

where $\bar{r}_x^{(d)}$ can be computed from the iterative expression of $D_x^{(d)}$. For each interior node x, if x is satisfied then

$$\begin{aligned} \mathrm{DecryptNode}(C, sk_{\mathcal{T},\langle i\rangle}, x) &= \prod_{x' \in S_x} F_{x'}^{\Delta_{j,S_x'}(0)} \\ &= \prod_{x' \in S_x} \big(e(g, g_2)^{s \cdot q_{x'}^{(d)}(0)}\big)^{\Delta_{j,S_x'}(0)} \\ &= \prod_{x' \in S_x} \big(e(g, g_2)^{s \cdot q_{\mathrm{parent}(x')}^{(d)}(\mathrm{index}(x'))}\big)^{\Delta_{j,S_x'}(0)} \\ &= \prod_{x' \in S_x} e(g, g_2)^{s \cdot q_x^{(d)}(j) \cdot \Delta_{j,S_x'}(0)} \\ &= e(g, g_2)^{s \cdot q_x^{(d)}(0)}. \end{aligned}$$

We have $\mathrm{DecryptNode}(C, sk_{\mathcal{T},\langle i\rangle}, z) = e(g, g_2)^{s \cdot y} = e(g_1, g_2)^s$ if and only if $\mathcal{T}_{\langle i\rangle}\big(\gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}\big) = 1$. Thus,

$$\frac{C'}{\mathrm{DecryptNode}(C, sk_{\mathcal{T},\langle i\rangle}, z)} = \frac{me(g_1, g_2)^s}{e(g_1, g_2)^s} = m$$

if and only if $\mathcal{T}(\gamma) = 1$. Decryption succeeds.

B Proof of fs-ABE Security

Suppose there exists a polynomial-time adversary $\mathcal{A}$, that can attack the fs-ABE scheme in the selective chosen plaintext, a set of attributes and time period model with advantage ϵ. We construct a simulator $\mathcal{B}$ that can attack the GPSW-ABE scheme in the selective chosen plaintext, a set of attributes model with the same advantage ϵ. The simulation proceeds as follows:

We first let the challenger set the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with an efficient bilinear map e and generator g. Then the challenger sets the parameters of GPSW-ABE, and passes $g_1(= g^y), g_2, \tilde{t}_1, \tilde{t}_2, \cdots, \tilde{t}_{n+d+1}$ to $\mathcal{B}$. It keeps y as the master key.

Init The simulator $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{A}$ sends a set γ of attributes and a time period i on which it wishes to be challenged to $\mathcal{B}$. Then $\mathcal{B}$ sends the set $\gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}$ of attributes as its challenge set to the challenger.

Setup $\mathcal{B}$ chooses a random $n+d$ degree polynomial $f(X)$ and calculates a $n+d$ degree polynomial $u(X)$ as follows: set $u(X) = -X^{n+d}$ for all $X \in \gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}$ and $u(X) \neq -X^{n+d}$ for other X. $\mathcal{B}$ sets $t_j = g_2^{u(j)} g^{f(j)}$ for all $j = 1$ to $n+d+1$. Because $f(X)$ is a random $n+d$ degree polynomial, all t_j will be chosen independently at random as in the fs-ABE construction. Implicitly, $\mathcal{B}$ defines $T(X) = g_2^{X^{n+d}+u(X)} g^{f(X)}$. $\mathcal{B}$ passes $\{g_1, g_2, t_1, t_2, \cdots, t_{n+d+1}\}$ to $\mathcal{A}$ as the public parameters of fs-ABE.

Phase 1 $\mathcal{A}$ issues queries for private keys $SK_{\mathcal{T},j}$ associated with a tree $\mathcal{T}$ and a time period j. No private key query for the access tree $\mathcal{T}$ such that $\mathcal{T}(\gamma) = 1$ for any time $j \leq i$ is allowed. $\mathcal{B}$ has to generate the private keys for $\mathcal{A}$.

We first define a procedure PolyUnsat$(\mathcal{T}'|_x, \gamma', g^{\lambda_x})$, where $\mathcal{T}'|_x$ is an unsatisfied access tree with the root node x, γ' is a set of attributes such that $\mathcal{T}'|_x(\gamma') = 0$, and $\lambda_x \in \mathbb{Z}_p$. The procedure sets up a polynomial for each node of $\mathcal{T}'|_x$.

PolyUnsat$(\mathcal{T}'|_x, \gamma', g^{\lambda_x})$ It first defines a polynomial q_x of degree d_x for the root node x such that $q_x(0) = \lambda_x$ by setting $g^{q_x(0)} = g^{\lambda_x}$. For each x' of $h_x(\leq d_x)$ satisfied children of x, the procedure defines $q_{x'}(0) = \lambda_{x'}$, where $\lambda_{x'} \in \mathbb{Z}_p$ is chosen randomly, and sets $q_x(\text{index}(x')) = \lambda_{x'}$. To completely define q_x it then sets $d_x - h_x$ points of q_x randomly. For each subtree $\mathcal{T}'|_{x'}$ of $\mathcal{T}'|_x$, the algorithm proceeds as follows:

- If $\mathcal{T}'|_{x'}$ is satisfied, it sets $d_{x'}$ other points of $q_{x'}$ randomly to completely define $q_{x'}$. For any other node y in $\mathcal{T}'|_{x'}$, it completely defines q_y by setting $q_y(0) = q_{\mathrm{parent}(y)}(\mathrm{index}(y))$ and d_y other points randomly.
- If $\mathcal{T}'|_{x'}$ is not satisfied, it recursively calls $\mathrm{PolyUnsat}(\mathcal{T}'|_{x'}, \gamma', g^{q_x(\mathrm{index}(x'))})$. Notice that only $g^{q_x(\mathrm{index}(x'))}$ can be got by interpolation.

Notice that all defined polynomials satisfy to the constraint in the Key Generation algorithm of fs-ABE scheme.

For the query for $SK_{\mathcal{T},j}$ from $\mathcal{A}$, $\mathcal{B}$ distinguishes two cases:

- $\mathcal{T}(\gamma) = 1$ and $j > i$. In the construction of fs-ABE,

$$SK_{\mathcal{T},j} = \big(sk_{\mathcal{T},\langle j\rangle}, \{sk_{\mathcal{T},j_0 j_1 \cdots j_{k-1}1}\}_{j_k=0}\big).$$

$\mathcal{B}$ has to generate the secret keys associated with the trees $\mathcal{T}_{\langle j\rangle}$ and $\{\mathcal{T}_{j_0 j_1 \cdots j_{k-1}1}\}_{j_k=0}$. Since $j > i$, $\{j_0 j_1 \cdots j_{k-1}1\}_{j_k=0}$ are not prefixes of i. Thus,

$$\mathcal{T}_{j_0 j_1 \cdots j_{k-1}1}\big(\gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}\big) = 0.$$

The simulator $\mathcal{B}$ runs PolyUnsat on the trees $\mathcal{T}_{\langle j\rangle}$ and $\{\mathcal{T}_{j_0 j_1 \cdots j_{k-1}1}\}_{j_k=0}$ with $\big(\gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}, g_1\big)$. This defines a polynomial for each node of $\mathcal{T}_{\langle j\rangle}$ and $\{\mathcal{T}_{j_0 j_1 \cdots j_{k-1}1}\}_{j_k=0}$. The constant terms in the polynomials associated with the roots of the above trees are all y. $\mathcal{B}$ first constructs the secret key $sk_{\mathcal{T},\langle j\rangle}$ associated with the tree $\mathcal{T}_{\langle j\rangle}$ as follows. For each leaf node x of $\mathcal{T}_{\langle j\rangle}$,

- If $\mathrm{att}(x) \in \gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}$. $\mathcal{B}$ knows the polynomial q_x completely.

$$\begin{aligned} D_x^{(d)} &= g_2^{q_x(0)} T(\mathrm{att}(x))^{r_x^{(d)}} = g_2^{Q_x^{(d)}(0)} T(\mathrm{att}(x))^{r_x^{(d)}} \\ R_x^{(d)} &= g^{r_x^{(d)}} \end{aligned}$$

where $r_x^{(d)} \in \mathbb{Z}_p$ is chosen randomly, and we denote $q_x(0)$ by $Q_x^{(d)}(0)$ for consistency.

– If $k = \mathrm{att}(x) \notin \gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}$. $\mathcal{B}$ knows $g^{q_x(0)}$.

$$\begin{aligned}
D_x^{(d)} &= g^{\frac{-q_x(0)f(k)}{k^{n+d}+u(k)}}(g_2^{k^{n+d}+u(k)}g^{f(k)})^{r_x} \\
&= g_2^{q_x(0)}(g_2^{k^{n+d}+u(k)}g^{f(k)})^{\frac{-q_x(0)}{k^{n+d}+u(k)}}(g_2^{k^{n+d}+u(k)}g^{f(k)})^{r_x} \\
&= g_2^{q_x(0)}(g_2^{k^{n+d}+u(k)}g^{f(k)})^{r_x-\frac{q_x(0)}{k^{n+d}+u(k)}} \\
&= g_2^{q_x(0)}T(k)^{r_x^{(d)}} \\
&= g_2^{Q_x^{(d)}(0)}T(\mathrm{att}(x))^{r_x^{(d)}} \\
R_x^{(d)} &= g^{\frac{-q_x(0)}{k^{n+d}+u(k)}}g^{r_x} = g^{r_x-\frac{q_x(0)}{k^{n+d}+u(k)}} = g^{r_x^{(d)}}
\end{aligned}$$

where $r_x \in \mathbb{Z}_p$ is chosen randomly, $r_x^{(d)} = r_x - \frac{q_x(0)}{k^{n+d}+u(k)}$, and we denote $q_x(0)$ by $Q_x^{(d)}(0)$ for consistency.

$\mathcal{B}$ takes the set of above secret pairs as $sk_{\mathcal{T},\langle j\rangle}$. In a similar manner $\mathcal{B}$ can construct $\{sk_{\mathcal{T},j_0j_1\cdots j_{k-1}1}\}_{j_k=0}$ associated with the trees $\{\mathcal{T}_{j_0j_1\cdots j_{k-1}1}\}_{j_k=0}$. The distribution of each one of these secret keys is identical to that of the fs-ABE scheme. $\mathcal{B}$ passes these keys to $\mathcal{A}$ as $SK_{\mathcal{T},j}$.

- $\mathcal{T}(\gamma) = 0$. $\mathcal{B}$ runs PolyUnsat on the tree $\mathcal{T}$ with $(\mathcal{T}, \gamma, g_1)$. This defines a polynomial for each node of $\mathcal{T}$. The constant term in the polynomial associated with the root of $\mathcal{T}$ is y. Define the secret key associated with the tree $\mathcal{T}$ as in above case. $\mathcal{B}$ uses this key to recursively apply the Compute Next algorithm of fs-ABE, and obtains $SK'_{\mathcal{T},0}$. Then $\mathcal{B}$ runs the Update of fs-ABE with $SK'_{\mathcal{T},0}$ to generate $SK'_{\mathcal{T},j}$. The distribution of $SK'_{\mathcal{T},j}$ is identical to $SK_{\mathcal{T},j}$ in the fs-ABE scheme. $\mathcal{B}$ passes this key to $\mathcal{A}$ as $SK_{\mathcal{T},j}$.

Challenge The adversary $\mathcal{A}$ submits two equal messages m_0 and m_1 to the simulator $\mathcal{B}$. $\mathcal{B}$ submits m_0 and m_1 to the challenger. The challenger flips a fair coin $b \in \{0, 1\}$, and returns an encryption of m_b in the GPSW-ABE scheme. The ciphertext is output as:

$$\left(\gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}, m_b e(g_1, g_2)^s, g^s, \{\widetilde{T}(k)^s\}_{k\in\gamma\cup\{\text{“1-}i_1\text{”},\text{“2-}i_2\text{”},\cdots,\text{“d-}i_d\text{”}\}}\right)$$

$\mathcal{B}$ modifies this ciphertext as follows:

$$\left\langle i, \left(\gamma \cup \{\text{“1-}i_1\text{”}, \text{“2-}i_2\text{”}, \cdots, \text{“d-}i_d\text{”}\}, m_b e(g_1, g_2)^s, g^s, \{(g^s)^{f(k)}\}_{k\in\gamma\cup\{\text{“1-}i_1\text{”},\text{“2-}i_2\text{”},\cdots,\text{“d-}i_d\text{”}\}}\right)\right\rangle$$

Then $\mathcal{B}$ passes it to $\mathcal{A}$ as the challenge ciphertext, which is a valid random encryption of message m_b in the fs-ABE scheme since $(g^s)^{f(k)} = (g^{f(k)})^s = T(k)^s$ by the construction of $T(X)$.

Phase 2 The simulator $\mathcal{B}$ acts exactly as it did in Phase 1.

Guess The adversary $\mathcal{A}$ submits a guess b' of b to $\mathcal{B}$. $\mathcal{B}$ submits b' to the challenger as its guess of b.

As shown above the simulator's generation of public parameters, private keys and the challenge ciphertext is identical to that of the actual fs-ABE scheme. The advantage of the simulator attacking GPSW-ABE scheme in selective chosen plaintext, a set of attributes game is

$$\begin{aligned} & \left|\Pr[\mathcal{B}\text{'s guess of } b \text{ is correct}] - \frac{1}{2}\right| \\ = \; & \left|\Pr[\mathcal{A}\text{'s guess of } b \text{ is correct}] - \frac{1}{2}\right| \\ = \; & \epsilon. \end{aligned}$$

For reducing the security of fs-ABE to hardness of DBDH problem, we need the following Theorem 3 from [10].

Theorem 3 *If an adversary has advantage ϵ to attack the GPSW-ABE scheme in the selective chosen plaintext, a set of attributes model, then a simulator can be constructed to solve the DBDH problem with the advantage $\frac{1}{2}\epsilon$.*

Combining Theorem 1 and Theorem 3, we obtain the following Theorem 4 for the fs-ABE scheme.

Theorem 4 *If an adversary can break the fs-ABE scheme in the selective chosen plaintext, a set of attributes and time period model, then a simulator can be constructed to solve the DBDH problem with a non-negligible advantage.*

References

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

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

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

[4] 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.

[5] Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *ACM Conference on Computer and Communications Security*, pages 354–363, 2004.

[6] 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.

[7] 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.

[8] Tal Malkin, Daniele Micciancio, and Sara 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.

[9] Huijun Xiong, Xinwen Zhang, Wei Zhu, and Danfeng Yao. Cloudseal: End-to-end content protection in cloud-based storage and delivery services. In *SecureComm*, 2011.

[10] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

[11] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security*, pages 195–203, 2007.

[12] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[13] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[14] I3P. Attribute-based encryption in future applications. Website, 2009. `http://www.thei3p.org/research/attribute_encrypt.html`.

[15] Joseph A. Akinyele, Christoph U. Lehmann, Matthew D. Green, Matthew W. Pagano, Zachary N. J. Peterson, and Aviel D. Rubin. Self-protecting electronic medical records using attribute-based encryption. Cryptology ePrint Archive, Report 2010/565, 2010. `http://eprint.iacr.org/`.

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

[17] 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.

[18] 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.

[19] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[20] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, pages 213–229, 2001.

[21] Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[22] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.

[23] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, pages 619–636, 2009.

[24] Robert W. Bradshaw, Jason E. Holt, and Kent E. Seamons. Concealing complex policies with hidden credentials. In *ACM Conference on Computer and Communications Security*, pages 146–157, 2004.

[25] Saman Zarandioon, Danfeng Yao, and Vinod Ganapathy. K2c: Cryptographic cloud storage with lazy revocation and anonymous access. In *SecureComm*, 2011.

[26] Ming Li, Shucheng Yu, Ning Cao, and Wenjing Lou. Authorized private keyword search over encrypted personal health records in cloud computing. In *IEEE ICDCS*, 2011.

[27] Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *INFOCOM*, pages 525–533, 2010.

[28] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM*, pages 534–542, 2010.

[29] Jinshu Su, Dan Cao, Xiaofeng Wang, Yipin Sun, and Qiaolin Hu. Attribute-based encryption schemes. In *Journal of Software*, pages 1299–1315, 2011.

[30] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.