Flexible Attribute-Based Encryption

Seiko Arita *

2012.07.09
Graduate School of Information Security,
Institute of Information Security, Japan

Abstract. *In this paper, we propose a notion of* flexible attribute-based encryption. *Flexible attribute-based encryption is a variant of ciphertext-policy ABE, which allows one to* loosen *a decryption policy underlying a given ciphertext, if one knows some system-wide trapdoor information, without knowing its underlying plaintext message. We give a concrete construction of the flexible attribute-based encryption that satisfies indistinguishability under the loosening operation, based on the construction of ciphertext-policy ABE given by Bethencourt, Sahai and Waters.*

keywords: Attribute-based encryption, Ciphertext-policy, Loosening operation.

1 Introduction

A notion of attribute-based encryption (ABE) was first proposed by Sahai and Waters [17], in which, a message m is encrypted to a ciphertext c under some predicate f, and a user with credential X can decrypt the ciphertext c if and only if the predicate f is satisfied by the user's credential X: $f(X) = 1$. The concept of ABE was further clarified by Goyal, Pandey, Sahai, and Waters [8]. They proposed two complementary forms of ABE: Key-Policy ABE and Ciphertext-Policy ABE. In this paper, we focus on Ciphertext-Policy ABE, in which attributes are used to describe users' credentials and formulas over these attributes are attached to the ciphertext by the encrypting party.

The first construction of Ciphertext-Policy ABE was given by Bethencourt, Sahai, and Waters [4]. Its security is proved under the generic bilinear group with random oracle model. (We call the model which uses both the generic bilinear group and random oracle the generic bilinear group with random oracle model.) Following [4], many efforts have been done to improve the proof of security. Waters [20] gives a construction of ABE which can be proved under the standard model in a selective manner. Lewko, Okamoto, Sahai, Takashima, and Waters [13] and Okamoto and Takashima [15] give fully secure constructions of ABE in the standard model.

On a while, ABE has been applied in building a variety of secure systems [16, 19, 5, 1, 11]. One of major problems in these applications is that ABE-based systems tend to lack flexibility. A ciphertext once produced under decryption policy f never can be decrypted under a more loosened policy $\mathbf{or}(f, \Delta f)$ (if f is not satisfied) by the definition of security of ABE (of course). However, in reality, the degree of privacy of information is never fixed: yesterday's secret is not necessary secret of today. Even if some information is very restrictive (described as policy f) to be accessed at this moment of time, the same information gradually can be made more and more accessible (described as policy $\mathbf{or}(f, \Delta f)$) as time goes by.

* Supported by MEXT-Supported Program for the Strategic Research Foundation at Private Universities, 2011-2013.

Our contribution. In this paper, we propose a notion of *flexible attribute-based encryption*. Flexible attribute-based encryption is a variant of ciphertext-policy ABE, which allows one to *loosen* a decryption policy underlying a given ciphertext, if one knows some system-wide trapdoor information, without knowing its underlying plaintext message. More precisely, suppose a given ciphertext c was generated by encrypting a plaintext m under a decryption policy f. The flexible attribute-based encryption enables a "loosening operation" that, given Δf and some system-wide trapdoor information γ, converts the ciphertext c into a more unrestrictive version of ciphertext c' which encrypts the same plaintext m under the loosened policy $\mathbf{or}(f, \Delta f)$, without knowing the message m itself. Users having attributes that satisfy (only) the appended policy Δf now can decrypt the ciphertext c' to know the message m. Here we note that the trapdoor information γ is independent of individual policies or ciphertexts.

As one of applications of such flexible attribute-based encryption, we can consider an integration of cloud storage services. Suppose two storage services A and B are going to integrate into one storage service. Suppose, by policy mapping, that encrypted files C_{f_A} under policy f_A in service A now should be decrypted also by entities satisfying policy f_B in service B. The authenticated operator in service A with trapdoor γ can use the loosening operation against those C_{f_A} to get new encrypted files $C_{\mathbf{or}(f_A,f_B)}$ that can be decrypted also by entities satisfying policy f_B in service B.

We will see that there is a subtlety over security concerning such loosening operations and then we will define two notions of security of flexible attribute-based encryption, *indistinguishability under loosening operation* and *indistinguishability under loosening key*.

We also give a concrete construction of the flexible attribute-based encryption that satisfies the indistinguishability under loosening operation and the indistinguishability under loosening key, based on the construction of ciphertext-policy attribute-based encryption given by Bethencourt, Sahai and Waters [4]. Its security proof is given in the generic bilinear group with random oracle model.

Related works. The concept of our flexible attribute-based encryption is similar to the attribute-based proxy re-encryption [9, 12, 10, 7].

In the attribute-base proxy re-encryption, one can generate re-encryption key $rk_{f_1 \to f_2}$, and by using the key $rk_{f_1 \to f_2}$, a ciphertext c_{f_1} for policy f_1 can be re-encrypted into a ciphertext c_{f_2} for policy f_2. To generate such re-encryption key $rk_{f_1 \to f_2}$, the secret key sk_{f_1} for policy f_1 is required. On a while, in our flexible ABE, all ciphertexts can be "loosened" using the single (system-wide) trapdoor information γ (which is independent of individual policies).

The ABE scheme of [7] is more similar to our flexible attribute-based encryption in a sense. A user in the scheme of [7] can provide the proxy with a single transformation key that allows the proxy to translate any ABE ciphertext into a ElGamal-style ciphertext, that the user is able to decrypt under the user's secret key (with one simple exponentiation).

2 A Notion of Flexible Attribute-Based Encryption

A flexible attribute-based encryption scheme is a tuple of five PPT algorithms Setup, Enc, Ext, Dec and Loosen.

- $(par, mk, lk) \leftarrow \mathsf{Setup}(1^k)$.
  Algorithm Setup generates a public parameter par, a master secret mk and a trapdoor information lk for loosening, given a security parameter 1^k.

– $(f, c) \leftarrow \mathsf{Enc}(par, f, m)$.
Algorithm Enc encrypts a given message m to a ciphertext c under a given decryption policy represented as a boolean formula f.

– $(as, d) \leftarrow \mathsf{Ext}(par, mk, as)$.
Algorithm Ext generates a secret key d for a given attribute set as, using the master secret mk.

– $m \leftarrow \mathsf{Dec}(par, (f, c), (as, d))$.
Algorithm Dec decrypts a ciphertext (f, c) by using a secret key d for an attribute set as to obtain a resulting plaintext m. The plaintext m may be a special symbol $\perp$ indicating a decryption error if something is wrong.

– $(\mathrm{or}(f, \Delta f), c') \leftarrow \mathsf{Loosen}(par, lk, (f, c), \Delta f)$.
By using the dedicated trapdoor information lk, algorithm Loosen loosens a decryption policy of a given ciphertext (f, c) so that more entities, that satisfy some added policy Δf, can also decrypt the ciphertext c, resulting a new ciphertext $(\mathrm{or}(f, \Delta f), c')$.

Correctness requirement. Under any valid setup information (par, mk, lk) $(\leftarrow \mathsf{Setup}(1^k))$, if one encrypts any message $m \in \mathrm{Message}(k)$ under any decryption policy $f \in \mathrm{Policy}(k)$ to a ciphertext (f, c), then it must be decrypted to the original plaintext m as

$$\mathsf{Dec}(par, (f, c), (as, d)) \;=\; m,$$

if the secret key (as, d) $(\leftarrow \mathsf{Ext}(par, mk, as))$ is generated for some attribute set as that satisfies the decryption policy f.

If the ciphertext (f, c) is loosened by a policy Δf to a new ciphertext $(\mathrm{or}(f, \Delta f), c')$ as

$$(\mathrm{or}(f, \Delta f), c') \;\leftarrow\; \mathsf{Loosen}(par, lk, (f, c), \Delta f),$$

then the resulting ciphertext c' must be decrypted to the original plaintext m as

$$\mathsf{Dec}(par, (\mathrm{or}(f, \Delta f), c'), (as', d')) \;=\; m,$$

even if the attribute set as' satisfies the appended policy Δf (or f).

Regarding security under loosening operations. Before defining security in a formal way, here we consider some aspects regarding security of such attribute-based encryption that gives loosening operations to users.

First of all, the loosening operation should be performed by some entity with possession of the trapdoor information lk without knowing the underlying message. This will be captured in the security condition named 'indistinguishability under loosening key'.

Another point is a more subtle one. Suppose an adversary A obtains a ciphertext c^* of a plaintext m under a policy $f = A$ or B or C. It is plausible that A manages to construct a ciphertext c' of the same plaintext m (without knowing m itself) under a more restricted policy $f' = A$ or B, based on the ciphertext c^*. Then, A can use the loosening operation on c' to get another ciphertext c'' also of the same plaintext m but under a loosened policy $f'' = A$ or B or D and then A could know the underlying plaintext m of the original ciphertext c^* by using a corrupt key d_D of the added attribute D against the ciphertext c''.

That scenario means that a victim's ciphertext c^* can be corrupted even if c^* itself has never been processed under loosening operations. (Off course, if the attribute-based encryption has CCA-security, that type of attack based on malleability can be avoided. However, at the same time we lose the loosening operations, too.)

We will require that loosening operations for c' different from c^* should never affect the security of c^*, in the security condition named 'indistinguishability under loosening operation'.

3 Security of Flexible Attribute-Based Encryption

To define security of a flexible attribute encryption scheme, we describe games using the notation of the framework of code-based games [3]. In the framework, a game has an **Initialize** procedure, procedures to respond to adversary oracle queries, and a **Finalize** procedure. A game $\mathbf{Game_A}$ is executed with an adversary A as follows. First, **Initialize** executes, and its outputs are the inputs to A. Then A executes, its oracle queries being answered by the corresponding procedures of $\mathbf{Game_A}$. When A terminates, its output becomes the input to the **Finalize** procedure. The output of the latter is called the output of the game, and we let $y \leftarrow \mathbf{Game}_A$ denote the event that this game output takes value y.

3.1 Indistinguishability under Loosening Operation

Let $\mathsf{FABE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Ext}, \mathsf{Dec}, \mathsf{Loosen})$ be a flexible attribute encryption scheme. Let A be an arbitrary PPT adversary against FABE. In our game $\mathbf{Game}^{ind-lso}_{A,\mathsf{FABE}}(k)$, procedures **Initialize** and **Finalize** are defined as follows.

- **procedure Initialize**:
  - $b \xleftarrow{\$} \{0,1\}$
  - $(par, mk, lk) \leftarrow \mathsf{Setup}(1^k)$
  - return par.
- **procedure Finalize** (b'):
  - return $b' \stackrel{?}{=} b$.

Following procedures **Extract**, **LR** and **Loosen** are used in $\mathbf{Game}^{ind-lso}_{A,\mathsf{FABE}}(k)$ to answer oracles queries from A:

- **procedure Extract** (as):
  - assert$(f^*(as) = \mathsf{false})$
  - $(as, d) \leftarrow \mathsf{Ext}(par, mk, as)$
  - return (as, d).
- **procedure LR** (f^*, m_0, m_1):
  - assert$(f^*(as) = \mathsf{false})$ for as's submitted to **Extract**-oracle.
  - $(f^*, c^*) \leftarrow \mathsf{Enc}(par, f^*, m_b)$
  - return (f^*, c^*).
- **procedure Loosen** $((f, c), \Delta f)$:
  - assert$((f, c) != (f^*, c^*))$
  - $(f', c') \leftarrow \mathsf{Loosen}(par, lk, (f, c), \Delta f)$
  - return (f', c').

In the above, "assert($f^*(as) = \mathsf{false}$)" means that one must check whether the condition $f^*(as) = \mathsf{false}$ holds or not if f^* already defined, and abort if it does not hold, or else continue. Similar for "assert($(f, c) != (f^*, c^*)$)".

Definition 1. *A flexible attribute encryption scheme* FABE *is said to be* indistinguishable under loosening operation (IND-LSO) *if for an arbitrary PPT adversary A its advantage* $\mathbf{Adv}^{ind-lso}_{A,\mathsf{FABE}}(k) := |\Pr[\mathrm{Game}^{ind-lso}_{A,\mathsf{FABE}}(k) = 1] - 1/2|$ *is a negligible function in k.*

3.2 Indistinguishability under Loosening Key

Let $\mathsf{FABE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Ext}, \mathsf{Dec}, \mathsf{Loosen})$ be a flexible attribute encryption scheme. Let A be an arbitrary PPT adversary against FABE. In our game $\mathbf{Game}^{ind-lsk}_{A,\mathsf{FABE}}(k)$, procedures **Initialize** and **Finalize** are defined as follows.

- **procedure Initialize**:
  - $b \xleftarrow{\$} \{0,1\}$
  - $(par, mk, lk) \leftarrow \mathsf{Setup}(1^k)$
  - return (par, lk).
- **procedure Finalize** (b'):
  - return $b' \stackrel{?}{=} b$.

Here, we note that **Initialize** returns a trapdoor information lk for loosening operation as well as parameter par (and adversaries A will know lk as well as par).

Procedure **LR** is used in $\mathbf{Game}^{ind-lsk}_{A,\mathsf{FABE}}(k)$ to answer oracle queries from A:

- **procedure LR** (f^*, m_0, m_1):
  - $(f^*, c^*) \leftarrow \mathsf{Enc}(par, f^*, m_b)$
  - return (f^*, c^*).

Definition 2. *A flexible attribute encryption scheme* FABE *is said to be* indistinguishable under loosening key (IND-LSK) *if for an arbitrary PPT adversary A its advantage* $\mathbf{Adv}^{ind-lsk}_{A,\mathsf{FABE}}(k) := |\Pr[\mathrm{Game}^{ind-lsk}_{A,\mathsf{FABE}}(k) = 1] - 1/2|$ *is a negligible function in k.*

We note that adversary A has no access to **Extract**-oracle in the game $\mathrm{Game}^{ind-lsk}$. Since A has loosening key lk, A can trivially decrypt the challenge ciphertext if A has access to **Extract**-oracle.

4 Concrete Scheme

We show a concrete flexible attribute encryption scheme. We start with a notion of linear-dependency resistant function, which is used as a building block for the scheme.

4.1 Linear-Dependency Resistant Function

Definition 3. *A function $F : \{0,1\}^* \to \{0,1\}^{2N}$ is said to be* N-linear-dependency resistant *if no PPT algorithm A is not able to generate n distinct strings $x_1, \ldots, x_n$ satisfying that $F(x_1), \ldots, F(x_n)$ are linearly dependent as vectors over Z_2 except with a negligible probability for any $n \leq N$.*

It is easy to see that statistically pseudorandom function F with range $\{0,1\}^{2N}$ is in fact N-linear-dependency resistant. So, a hash function $F : \{0,1\}^* \to \{0,1\}^{2N}$ is N-linear-dependency resistant in the random oracle model [2] with respect to F.

4.2 The FABE scheme

We construct a concrete flexible attribute encryption scheme based on the attribute encryption scheme of Bethencourt, Sahai and Waters [4].

In the followings, $\sharp\mathrm{Leaf}(f)$ denotes a number of leaf nodes of a given binary formula f. $(\rho, M) \leftarrow \mathrm{LSS}(p, f)$ denotes a transformation to convert a boolean formula f into a linear secret sharing scheme defined by a share-generating matrix M over prime p (with corresponding secret-restoring coefficients $(\omega_i)_i$) with an assignment function ρ from the rows of matrix M to the universe of attributes. For its details we refer to [14]. Predicate $\mathrm{IsDH}(g, g_1, g_2, g_3)$ means the tuple (g, g_1, g_2, g_3) is a Diffie-Hellman tuple, i.e., $g_3 = g_2^a$ for a satisfying $g_1 = g^a$. For vectors $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_n)$, their inner product is written as $a \cdot b = \sum_{i=1,\ldots,n} a_i b_i$.

- Setup $(1^k, N(k))$:
  - Generate a bilinear group parameter $(g, p, e) \leftarrow \mathsf{GenGrp}(1^k)$.
  - Select a pseudorandom function $F\ (= F_1 \cdots F_{2N}) : \{0,1\}^* \to \{0,1\}^{2N}$.
  - Choose $\alpha, \beta, \gamma_1, \ldots, \gamma_{2N} \xleftarrow{\$} Z_p$.
  - Compute $h = g^\beta, f = g^{1/\beta}, w = e(g,g)^\alpha, u_1 = g^{\gamma_1}, \ldots, u_{2N} = g^{\gamma_{2N}}$.
  - Return $par = (g, h, f, w, F, u_1, \ldots, u_{2N})$, $mk = (\beta, g^\alpha)$ and $lk = \gamma := (\gamma_1, \ldots, \gamma_{2N})$.
    `/* Elements` $(u_1, \ldots, u_{2N})$ `defines a hash function` $\mathcal{H}(S) = u_1^{F_1(S)} \cdots u_{2N}^{F_{2N}(S)}$. `*/`

- Enc (par, f, m):
  - Assert $n := \sharp\mathrm{Leaf}(f) < N$.
  - Convert the boolean formula f into a Linear Secret Sharing (LSS) scheme : $(\rho, M) \leftarrow \mathrm{LSS}(p, f)$. `/* Let dimension of` M `be` $n \times l$ `*/`
  - Choose $s, r_2, \ldots, r_l \xleftarrow{\$} Z_p$ and compute $s_i = M_i \cdot (s, r_2, \ldots, r_l)$ for $i \in [1..n]$.
  - Compute $c_0 = mw^s$, $c_1 = g^s$, $c_2 = h^s$, $c_3 = (g^{s_i})_{i\in[1..n]}$, $c_4 = (\mathcal{H}(\rho(i))^{s_i})_{i\in[1..n]}$, and $c_5 = \mathcal{H}(f, c_0, \ldots, c_4)^s$.
  - Return $(f, c = (c_0, \ldots, c_5))$.

- Ext (f, mk, as):
  - Choose $r \xleftarrow{\$} Z_p$ and $r_a \xleftarrow{\$} Z_p$ for each $a \in as$.
  - Compute $d_1 = g^{(\alpha+r)/\beta}$, $d_2 = (g^r \mathcal{H}(a)^{r_a})_{a\in as}$, and $d_3 = (g^{r_a})_{a\in as}$.
  - Return $d = (as, d_1, d_2, d_3)$.

- Dec $(par, (f, c), (as, d))$:
  - Convert the boolean formula f into a LSS scheme : $(\rho, M) \leftarrow \mathrm{LSS}(p, f)$.
  - Compute $I = \rho^{-1}(as)$ and the corresponding secret-restoring coefficients $\{\omega_i\}_{i\in I}$.
  - Compute $\kappa = \prod_{i\in I}\{e(c_{3,i}, d_{2,\rho(i)})/e(c_{4,i}, d_{3,\rho(i)})\}^{\omega_i}$.
  - Return $\kappa c_0 / e(d_1, c_2)$.

- Loosen $(par, lk, (f, c), \Delta f)$:
  - Loosen the policy f to $f' = \mathrm{or}(f, \Delta f)$.
  - Assert $n' := \sharp\mathrm{Leaf}(f') < N$.
  - Assert $\mathrm{IsDH}(g, \mathcal{H}(f, c_0, \cdots, c_4), c_1, c_5)$.
  - Set $c'_0 = c_0$, $c'_1 = c_1$, $c'_2 = c_2$.
  - Convert the boolean formula f' into a LSS scheme : $(\rho, M) \leftarrow \mathrm{LSS}(p, f')$. `/* Let dimension of` M `be` $n' \times l$ `*/`

- Let $g^{s'} = c'_1$ and choose $r_2, \ldots, r_l \xleftarrow{\$} Z_p$. /* We don't know the value of s'. */
- Compute $g^{s'_i} = g^{M_i \cdot (s', r_2, \ldots, r_l)}$ for $i \in [1..n']$ and set $c'_3 = (g^{s'_i})_{i \in [1..n']}$. /* The knowledge $g^{s'}$ is enough to compute $g^{s'_i}$ in the above. */
- Using the loosening key $lk = \gamma$, compute $c'_4 = (g^{s'_i(\gamma \cdot F(\mathcal{H}(i)))})_{i \in [1..n']}$ and $c'_5 = (c'_1)^{\gamma \cdot F(f', c'_0, \ldots, c'_4)}$. /* $\gamma \cdot F(x)$ denotes $\sum_{i=1,\ldots,2N} F_i(x)\gamma_i$. */
- Return $c' = (f', c'_0, \ldots, c'_5)$.

The correctness of the FABE scheme can be verified in a straight-forward way.

4.3 Security of the FABE scheme

We can prove security of the FABE scheme, depending on the generic bilinear group with random oracle model [6, 18].

Theorem 1. *The FABE scheme with parameter $N = N(k)$ is indistinguishable under loosening operation in the generic bilinear group model, under the assumption that the function F is $(N+1)$-linear-dependency resistant, and in the constraint that encryptions are allowed only with respect to "small" policies f that satisfy $\sharp\mathrm{Leaf}(f) < N$.*

Theorem 2. *The FABE scheme is indistinguishable under loosening key in the generic bilinear group model.*

Thus, the scheme is both indistinguishable under loosening operation and indistinguishable under loosening key in the generic bilinear group (w.r.t. GenGrp) with random oracle (w.r.t. F) model.

4.4 Proof of Theorems

Random encoding. Let Ψ_0 and Ψ_1 be two independent random encoding functions from Z_p to $\{0,1\}^{3\log(p)}$ with ranges G_0 and G_1, respectively:

$$\Psi_0 \ : \ Z_p \to G_0 \ (\subset \{0,1\}^{3\log(p)}) \tag{1}$$
$$\Psi_1 \ : \ Z_p \to G_1 \ (\subset \{0,1\}^{3\log(p)}). \tag{2}$$

Let e be the pairing on $G_0 \times G_0$ to G_1 , that is induced from a bilinear function $(a, b) \mapsto ab$ on $Z_p \times Z_p$ to Z_p:

$$e(\Psi_0(a), \Psi_0(b)) \ = \ \Psi_1(ab).$$

In what follows, we implicitly assume that an adversary A always has access to oracles for group operations in G_0 and G_1 and an oracle for pairing computation e.

Notation. Suppose an adversary A is given some group elements $\{\Psi_0(\alpha_i)\}_{i=1,\ldots,m}$ in G_0 and $\{\Psi_1(\beta_j)\}_{j=1,\ldots,l}$ in G_1. (A does not know any of pre-images α_i nor β_j.) Here, we can assume no $\Psi_0(\alpha_i)$ are redundant, that is any $\Psi_0(\alpha_i)$ cannot be derived from other given elements as $\Psi_0(\alpha_i) = \sum_{k \neq i} c_k \Psi_0(\alpha_k)$ with known constants c_k to adversary A. Similar for $\Psi_1(\beta_j)$.

Starting from the given elements $\{\Psi_0(\alpha_i)\}_{i=1,\ldots,m}$ and $\{\Psi_1(\beta_j)\}_{j=1,\ldots,l}$, A can compute any linear combinations $\sum_{i=1,\ldots,m} a_i \Psi_0(\alpha_i) = \Psi_0(\sum_{i=1,\ldots,m} a_i \alpha_i)$ or $\sum_{j=1,\ldots,l} b_j \Psi_1(\beta_j) = \Psi_1(\sum_{j=1,\ldots,l} b_j \beta_j)$ for any constants a_i, b_j known to A, by using the group-operation oracles, as well as elements

$e(c_i\Psi_0(\alpha_i), c_j\Psi_0(\alpha_j)) = \Psi_1(c_i\alpha_i, c_j\alpha_j)$ in G_1 for any constants c_i known to A by using the pairing-computation oracle. Thus, if A is once given elements $\{\Psi_0(\alpha_i)\}_{i=1,\ldots,m}$ and $\{\Psi_1(\beta_i)\}_{i=1,\ldots,l}$, its possible maximum view View_A can be described as a union of two vector spaces over Z_p:

$$\mathrm{View}_A = \langle \Psi_0(\alpha_i) \mid i \in [1..m] \rangle \ \cup \ \langle \Psi_1(\beta_j), \Psi_1(\alpha_s\alpha_t) \mid s, t \in [1..m], j \in [1..l] \rangle. \tag{3}$$

Here, we note that A is not able to sample elements in G_0 or G_1 by itself (except with a negligible probability) since those elements are encoded in a very redundant way into $\{0,1\}^{3\log(p)}$, which is exponentially larger than the size p of groups G_0 or G_1. Moreover, by a standard argument in the generic group model, we see that the elements $\{\Psi_0(\alpha_i)\}_{i=1,\ldots,m}$ in Equation (3) are linearly independent in the view of A. In fact, suppose A can compute some constants $\{c_i\}_{i=1,\ldots,m}$ satisfying $\sum_{i=1,\ldots,m} c_i\Psi_0(\alpha_i) = \Psi_0(0)$. Then $\sum_{i=1,\ldots,m} c_i\alpha_i = 0$, but this is impossible since the view of A is independent from those α_i's (A is given only their random encodings $\Psi_0(\alpha_i)$) and so the probability of such c_i are derived by A is bounded by a negligible probability $\Pr_{\alpha_1,\ldots,\alpha_m}[\sum_{i=1,\ldots,m} c_i\alpha_i = 0]$ with respect to the given constants $\{c_i\}$. Similarly, elements $\{\Psi_1(\beta_j)\}_{j=1,\ldots,l}$ and $\{\Psi_1(\alpha_s\alpha_t)\}_{s,t=1,\ldots,m}$ are linearly independent in the view of A.

To keep notation simple in View_A, hereafter, we omit the explicit description of Ψ_1-derived elements $\Psi_1(\alpha_s\alpha_t)$ (from $\Psi_0(\alpha_i)$'s by pairing computation) and merge Ψ_0-elements and Ψ_1-elements into a single space (even though sums of Ψ_0-element and Ψ_1-element are meaningless), so that View_A of Equation (3) will be denoted as:

$$\mathrm{View}_A = \langle \Psi_0(\alpha_i), \Psi_1(\beta_j), \text{derived-}\Psi_1\text{-elements} \mid i \in [1..m], j \in [1..l] \rangle.$$

Proof of Theorem 1

Simulation. We simulate the game $\mathbf{Game}^{ind-lso}_{A,\mathsf{FABE}}$ between an arbitrary adversary A and the proposed FABE scheme in the generic bilinear group model with the encoding functions Ψ_0 and Ψ_1. First, A is invoked on a parameter par that is computed by procedure $\mathbf{Initialize}(1^k)$. Then, A runs given oracle accesses to procedures $\mathbf{Extract}(\cdot)$, $\mathbf{LR}(\cdot,\cdot,\cdot)$ and $\mathbf{Loosen}(\cdot,\cdot)$ under the constraints described in $\mathbf{Game}^{ind-lso}_{A,\mathsf{FABE}}$. Finally, A's output is given to procedure $\mathbf{Finalize}(\cdot)$ to get a final result of the game. We need to show that the final result is equal to one only with a probability that is negligibly larger than one half.

The procedure **Initialize** is simulated honestly in accordance with Setup algorithm of the scheme as follows:

- **procedure Initialize** $(1^k, 1^{N(k)})$:
  - Choose $b \xleftarrow{\$} \{0,1\}$.
  - Choose $\alpha, \beta, \gamma_1, \ldots, \gamma_{2N} \leftarrow Z_p$.
  - Select a hash function $F(= F_1 \cdots F_{2N}) : \{0,1\}^* \to \{0,1\}^{2N}$.
  - Compute

$$g = \Psi_0(1), h = g^\beta = \Psi_0(\beta), f = g^{1/\beta} = \Psi_0(1/\beta),$$
$$u_1 = g^{\gamma_1} = \Psi_0(\gamma_1), \ldots, u_{2N} = g^{\gamma_{2N}} = \Psi_0(\gamma_{2N}), w = e(g,g)^\alpha = \Psi_1(\alpha).$$

  - Return $par = (g, h, f, w, F, \{u_i\}_{i=1,\ldots,2N})$.

  Given the output par, the (maximum) view of A is described as

$$\mathrm{View}_A \ = \ \langle \Psi_0(1), \Psi_0(\beta), \Psi_0(1/\beta), \Psi_0(\gamma_1), \ldots, \Psi_0(\gamma_{2N}), \Psi_1(\alpha), \text{derived-}\Psi_1\text{-elements} \rangle.$$

  The procedure **Extract** is also simulated honestly:

– **procedure Extract** (as_j)
  - Assert $f^*(as_j) = \mathsf{false}$.
  - Choose $r^{(j)} \xleftarrow{\$} Z_p$ and $r_a^{(j)} \xleftarrow{\$} Z_p$ for each $a \in as_j$.
  - Compute $d_1^{(j)} = g^{(\alpha+r^{(j)})/\beta} = \Psi_0((\alpha + r^{(j)})/\beta)$, $d_2^{(j)} = (g^{r^{(j)}}\mathcal{H}(a)^{r_a^{(j)}})_{a\in as_j} = (\Psi_0(r^{(j)} + r_a^{(j)}(F(a)\cdot\gamma)))_{a\in as_j}$, and $d_3^{(j)} = (g^{r_a^{(j)}})_{a\in as_j} = (\Psi_0(r_a^{(j)}))_{a\in as_j}$.
  - Return $(d_1^{(j)}, d_2^{(j)}, d_3^{(j)})$.

Receiving the result of **Extract**(as_j), following pieces of information are added to the view of A:

$$\text{View}_A \mathrel{+}= \langle \Psi_0((\alpha + r^{(j)})/\beta), \Psi_0(r^{(j)} + r_a^{(j)}(\gamma \cdot F(a))), \Psi_0(r_a^{(j)}), \text{derived-}\Psi_1\text{-elements} \mid a \in as_j \rangle.$$

The procedure **Loosen** is simulated honestly, too:

– **procedure Loosen** $((f^{(l)}, c^{(l)} = (c_0^{(l)}, \ldots, c_5^{(l)})), \Delta f)$:
  - Assert $(f^{(l)}, c^{(l)}) \neq (f^*, c^*)$.
  - Loosen the policy $f' = \text{or}(f^{(l)}, \Delta f)$. If $n' = \sharp\text{Leaf}(f') \geq N$, then return $\perp$.
  - Assert $\text{IsDH}(g, \mathcal{H}(f, c_0^{(l)}, \cdots, c_4^{(l)}), c_1^{(l)}, c_5^{(l)})$.
  - Set $c_0' = c_0$, $c_1' = c_1$, $c_2' = c_2$.
  - Convert the boolean formula f' into a LSS scheme : $(\rho, M) \leftarrow \text{LSS}(p, f')$.
    `/* Let dimension of` M `be` $n' \times l$ `*/`
  - Let $g^{\hat{s}} = c_1'$ (we don't know the value of $\hat{s}$) and choose $r_2, \ldots, r_l \xleftarrow{\$} Z_p$.
  - Compute $g^{\hat{s}_i} = g^{M_i \cdot (\hat{s}, r_2, \ldots, r_l)}$ for $i \in [1..n']$ and set $c_3' = (g^{\hat{s}_i} = \Psi_0(\hat{s}_i))_{i\in[1..n']}$.
  - Compute, using the key $lk = \gamma$, $c_4' = (g^{\hat{s}_i(\gamma\cdot F(\rho(i)))})_{i\in[1..n']} = (\Psi_0(\hat{s}_i(\gamma \cdot F(\rho(i)))))_{i\in[1..n']}$ and $c_5' = (c_1')^{\gamma\cdot F(f', c_0', \ldots, c_4')} = \Psi_0(\hat{s}(\gamma \cdot F(f', c_0', \ldots, c_4')))$.
  - Return $c' = (f', c_0', \ldots, c_5')$.

Receiving the result of **Loosen**$(f, c^{(l)} = (c_0^{(l)}, \ldots, c_5^{(l)}), \Delta f)$, following pieces of information are added to the view of A:

$$\text{View}_A \mathrel{+}= \langle \Psi_0(\hat{s}_i), \Psi_0(\hat{s}_i(\gamma \cdot F(\rho(i)))), \Psi_0(\hat{s}_i(\gamma \cdot F(f', c_0', \ldots, c_4'))), \text{derived-}\Psi_1\text{-elements} \mid i \in [1..n'] \rangle.$$

The procedure **LR** is simulated as follows:

– **procedure LR** (f^*, m_0, m_1):
  - Assert $n^* = \sharp\text{Leaf}(f^*) < N$.
  - Convert the boolean formula f^* into a LSS scheme : $(\rho, M) \leftarrow \text{LSS}(p, f^*)$.
    `/* Let dimension of` M `be` $n^* \times l$ `*/`
  - Choose $s^*, r_2, \ldots, r_l \xleftarrow{\$} Z_p$.
  - For $i \in [1..n^*]$, compute $s_i^* = M_i \cdot (s^*, r_2, \ldots, r_l)$.
  - Choose $\theta \xleftarrow{\$} Z_p$ and compute $c_0^* = e(g,g)^\theta = \Psi_1(\theta)$.
  - Compute $c_1^* = g^{s^*} = \Psi_0(s^*)$, $c_2^* = h^{s^*} = \Psi_0(\beta s^*)$, $c_3^* = (g^{s_i^*})_{i\in[1..n^*]} = (\Psi_0(s_i^*))_{i\in[1..n^*]}$, $c_4^* = (\mathcal{H}(\rho(i))^{s_i^*})_{i\in[1..n^*]} = (\Psi_0(s_i^*(\gamma \cdot F(\rho(i)))))_{i\in[1..n^*]}$, and $c_5^* = \mathcal{H}(f^*, c_0^*, \ldots, c_4^*)^{s^*} = \Psi_0(s^*(\gamma \cdot F(f^*, c_0^*, \ldots, c_4^*)))$.
  - Return $(f^*, c^* = (c_0^*, \ldots, c_5^*))$.

Receiving the result of **LR**(pl^*, m_0, m_1), following pieces of information are added to the view of A:

$$\begin{aligned}\text{View}_A \mathrel{+}= \langle &\Psi_1(\theta), \Psi_0(s^*), \Psi_0(\beta s^*), \Psi_0(s_i^*), \Psi_0(s_i^*(\gamma \cdot F(\rho(i)))), \Psi_0(s^*(\gamma \cdot F(f^*, c_0^*, \ldots, c_4^*))), \\ &\text{derived-}\Psi_1\text{-elements} \mid i \in [1..n^*] \rangle\end{aligned}$$

Analysis. The simulated **LR** procedure uses an independent fresh randomness θ to compute c_0^* (instead of αs^*). Hence the output of the simulated **LR**-oracle is completely independent from the choice of m_b, and the advantage of A in the simulated game is equal to 0. So, all we need to prove is that the simulated view of A is indistinguishable from its real view.

First we show the following lemma. [1]

Lemma 1. *Suppose adversary A submits no query to* **Loosen***-oracle. Then, the simulated view of A is statistically indistinguishable from the real view of A.*

Proof. The difference between the simulated and real views is only that the simulated **LR** procedure uses an independent fresh randomness θ to compute c_0^*, but the real **LR** procedure uses αs^* for that sake.

Under the generic bilinear-group model, distinct group elements are encoded in distinct independent random strings. So, to prove the lemma, it is sufficient that $\Psi_1(\alpha s^*)$ never appear in the simulated A's view View_A, provided that A submits no query to **Loosen**-oracle.

By the above observations, we know that the simulated view View_A is described as

$$\begin{aligned}\mathrm{View}_A = {} & \langle \Psi_0(1), \Psi_0(\beta), \Psi_0(1/\beta), \Psi_0(\gamma_1), \ldots, \Psi_0(\gamma_{2N}), \Psi_1(\alpha) \rangle \\ & + \sum_j \langle \Psi_0((\alpha + r^{(j)})/\beta), \Psi_0(r^{(j)} + r_a^{(j)}(\gamma \cdot F(a))), \Psi_0(r_a^{(j)}) \mid a \in as_j \rangle \\ & + \langle \Psi_1(\theta), \Psi_0(s^*), \Psi_0(\beta s^*), \Psi_0(s_i^*), \Psi_0(s_i^*(\gamma \cdot F(\rho(i)))), \Psi_0(s^*(\gamma \cdot F(f^*, c_0^*, \ldots, c_4^*))) \mid i \in [1..n^*] \rangle \\ & + \langle \text{derived-}\Psi_1\text{-elements} \rangle. \end{aligned} \tag{4}$$

Suppose, towards a contradiction, that $\Psi_1(\alpha s^*) \in \mathrm{View}_A$. Since $\Psi_1((\alpha + r^{(j)})s^*) = e(\Psi_0((\alpha + r^{(j)})/\beta), \Psi_0(\beta s^*)) \in \mathrm{View}_A$, this means that for any j, $\Psi_1(r^{(j)} s^*) \in \mathrm{View}_A$. However, because s^* is distributed among $\{s_i^*\}$, this is possible only if $\mathrm{View}_A \ni \Psi_1(r^{(j)} s_i^*)$ for plenty of indexes i in $[1..n^*]$. Any possible way to compute those $\Psi_1(r^{(j)} s_i^*)$ must be via elements $\Psi_0(r^{(j)} + r_a^{(j)}(\gamma \cdot F(a)))$ and $\Psi_0(s_i^*(\gamma \cdot F(\rho(i))))$ satisfying $a = \rho(i)$ for some $a \in as_j$. (In fact, the element $r^{(j)}$ exists only in $\Psi_0(r^{(j)} + r_a^{(j)}(\gamma \cdot F(a)))$ as a summand, and, after pairing it with $\Psi_0(s_i^*)$, in order to cancel out the effect of another summand $r_a^{(j)}(\gamma \cdot F(a))$ we need $\Psi_0(s_i^*(\gamma \cdot F(\rho(i))))$ satisfying $\rho(i) = a$. In the below, we call this $\Psi_0(s_i^*(\gamma \cdot F(\rho(i))))$ *critical information* on the share s_i^*.) This means that the query as_j includes attributes a satisfying $a = \rho(i)$ for plenty of shares s_i^*, that is, the target policy f^* is being satisfied by the query as_j, contradicting to the game constraint. □

Now, by Lemma 1 and its proof, all we need to show is that Loosen-oracle queries never help A to obtain the critical information on the shares s_i^* of the randomness s^* used to compute the challenge ciphertext c^*, i.e., the elements $\Psi_0(s_i^*(\gamma \cdot F(\rho(i))))$ for $i \in [1..n^*]$.

Suppose query $(f^{(l)}, c^{(l)} = (c_0^{(l)}, \ldots, c_5^{(l)}), \Delta f)$ is submitted to **Loosen**-oracle. Since the oracle returns a meaningful result only if $\mathrm{IsDH}(g, \mathcal{H}(f^{(l)}, c_0^{(l)}, \ldots, c_4^{(l)}), c_1^{(l)}, c_5^{(l)})$ holds, we can assume that there exists $\hat{s}$ satisfying

$$c_1^{(l)} = \Psi_0(\hat{s}),\ c_5^{(l)} = \Psi_0(\hat{s}(\gamma \cdot F(f^{(l)}, c_0^{(l)}, \ldots, c_4^{(l)}))).$$

This means that

$$\Psi_0(\hat{s}), \Psi_0(\hat{s}(\gamma \cdot F(f^{(l)}, c_0^{(l)}, \ldots, c_4^{(l)}))) \ \in\ \mathrm{View}_A$$

[1] Lemma 1 is essentially the same as Theorem 1 of [4]. But here we describe its proof for completeness and for later arguments.

already before A submitting the query and A will receive the answer $\Psi_0(\hat{s}_i)$, $\Psi_0(\hat{s}_i(\gamma \cdot F(\rho(i))))$ for $i \in [1..n']$ and $\Psi_0(\hat{s}(\gamma \cdot F(f', c'_0, \ldots, c'_4)))$.

Then we see that in order for A to obtain any information on the critical information $\Psi_0(s_i^*(\gamma \cdot F(\rho(i))))$ (for some $i \in [1..n^*]$) the $\hat{s}$ must contain cs^* or $c_i s_i^*$ as a summand with some constants c, c_i known to A. By equation (4), we can suppose that $\hat{s}$ is equal to s^* or s_i^* for some $i \in [1..n^*]$.

Thus the proof of the theorem is reduced to the following claim.

Claim. A is not able to issue a (meaningful) query to Loosen-oracle with $\hat{s} = s^*$ or $\hat{s} = s_i^*$ for some $i \in [1..n^*]$ except with a negligible probability.

Proof. Suppose, towards a contradiction, that A issues a query $(f^{(l)}, c^{(l)}, \Delta f)$ to Loosen-oracle with $\hat{s} = s_i^*$ for some $i \in [1..n^*]$. for which the oracle responds with some answer $\neq \perp$. (Here, we treat only with the case of $\hat{s} = s_i^*$. The case of $\hat{s} = s^*$ is similarly handled.)

Then, as seen above, we have

$$\Psi_0(s_i^*(\gamma \cdot F(f^{(l)}, c_0^{(l)}, \ldots, c_4^{(l)}))) \ \in \ \mathrm{View}_A \tag{5}$$

already before A submitting the Loosen-query. Since elements $(f^{(l)}, c^{(l)})$ in the query must not be equal to the target ciphertext (f^*, c^*) by the rule of game, we have $(f^{(l)}, c_0^{(l)}, \ldots, c_4^{(l)})$ is not equal to $(f^*, c_0^*, \ldots, c_4^*)$. Then, since $n^* < N$, by $(N+1)$-linear-dependency resistant property of F, we see that the vector $F(f^{(l)}, c_0^{(l)}, \ldots, c_4^{(l)})$ is linear independent of the vectors $\{F(\rho(i))\}_{i \in [1..n^*]} \cup \{F(f^*, c_0^*, \ldots, c_4^*)\}$ (except with a negligible probability). Hence, $\Psi_0(s_i^*(\gamma \cdot F(f^{(l)}, c_0^{(l)}, \ldots, c_4^{(l)})))$ cannot be a linear combination of vectors $\Psi_0(s_i^*(\gamma \cdot F(\rho(i))))$ (with $i \in [1..n^*]$) and $\Psi_0(s^*(\gamma \cdot F(f^*, c_0^*, \ldots, c_4^*)))$. But, this contradicts to Equation (5), because before submitting the Loosen-query all of the elements of the form $\Psi_0(s_i^*(\gamma \cdot F(\cdot)))$ that A knows are $\Psi_0(s_i^*(\gamma \cdot F(\rho(i))))$ (for $i \in [1..n^*]$) and $\Psi_0(s^*(\gamma \cdot F(f^*, c_0^*, \ldots, c_4^*)))$ from Equation (4). □

Proof of Theorem 2

Simulation. We simulate the game $\mathbf{Game}^{ind-lsk}_{A,\mathsf{FABE}}$ between an arbitrary adversary A and the proposed FABE scheme in the generic bilinear group model with the encoding functions Ψ_0 and Ψ_1. First, A is invoked on a parameter *par* and a loosening key γ that are computed by procedure $\mathbf{Initialize}(1^k)$. Then, A runs given oracle accesses to procedure $\mathbf{LR}(\cdot, \cdot, \cdot)$. Finally, A's output is given to procedure $\mathbf{Finalize}(\cdot)$ to get a final result of the game. We need to show that the final result is equal to one only with a probability that is negligibly larger than one half.

The procedure $\mathbf{Initialize}(1^k)$ and $\mathbf{LR}(\cdot, \cdot, \cdot)$ are simulated in the same way as in the game $\mathbf{Game}^{ind-lso}_{A,\mathsf{FABE}}$, except that procedure $\mathbf{Initialize}(1^k)$ now returns $\gamma = (\gamma_1, \ldots, \gamma_{2N})$ as well as *par*.

Analysis. As in the proof of Theorem 1, all we need to show is that $\Psi_1(\alpha s^*)$ never appear in the simulated A's view View_A, which is described as

$$\begin{aligned} \mathrm{View}_A = {} & \langle \Psi_0(1), \Psi_0(\beta), \Psi_0(1/\beta), \Psi_1(\alpha) \rangle_{\gamma_1, \ldots, \gamma_{2N}} \\ & + \langle \Psi_1(\theta), \Psi_0(s^*), \Psi_0(\beta s^*), \Psi_0(s_i^*) \mid i \in [1..n^*] \rangle_{\gamma_1, \ldots, \gamma_{2N}} \\ & + \langle \text{derived-}\Psi_1\text{-elements} \rangle_{\gamma_1, \ldots, \gamma_{2N}}. \end{aligned} \tag{6}$$

Here, we note that $(\gamma_1, \ldots, \gamma_{2N})$ are now known scalars to A. We emphasize it by giving them as indices of the right angle above, "$\rangle_{\gamma_1, \ldots, \gamma_{2N}}$".

By inspection of Equation (6), it is clear that View_A does not contain $\Psi_1(\alpha s^*)$. □

References

1. Randy Badenand, Adam Benderand, Neil Springand, Bobby Bhattacharjee, and Daniel Starin. "Persona: An online social network with user defined privacy", In ACM SIGCOMM, 2009.
2. M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols", ACM conference on Computer and Communications Security (ACM CCS), ppp. 62–73, 1993.
3. M. Bellare and P. Rogaway, "The Security of Triple Encryption and a Framework for Code- Based Game-Playing Proofs", EUROCRYPT 2006, LNCS 4004, pp. 409–426, Springer-Verlag, 2006.
4. John Bethencourt, Amit Sahai, Brent Waters, "Ciphertext-Policy Attribute-Based Encryption", SP '07, pp. 321–334, IEEE Computer Society, 2007.
5. Rakesh Bobba, Omid Fatemieh, Fariba Khan, Arindam Khanand Carl A. Gunter, Himanshu Khurana, and Manoj Prabhakaran, "Attribute-based messaging: Access control and confiden-tiality", ACM Transactions on Information and System Security (TISSEC), Volume 13 (4).
6. D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext", EUROCRYPT 2005, LNCS 3494, pp. 440–456, Springer, 2005.
7. M. Green, S. Hohenberger, B. Waters, "Outsourcing the Decryption of ABE Ciphertexts". In Usenix Security 2011.
8. 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.
9. S. Guo, Y. Zeng, J. Wei, and Q. Xu, "Attribute-based re-encryption scheme in the standard model", Wuhan University Journal of Natural Sciences, 13(5):621–625, 2008.
10. Luan Ibraimi, Muhammad Asim, Milan Petkovic, "An Encryption Scheme for a Secure Policy Updating", SECRYPT 2010, pp. 399–408, 2010.
11. Seny Kamara and Kristin Lauter, "Cryptographic Cloud Storage", FC 10, LNCS 6054, pp. 136–149, 2010.
12. X. Liang, Z. Cao, H. Lin, J. Shao, "Attribute Based Proxy Re-encryption with Delegating Capabilities", ASIACCS 09, pp. 276–286, 2009
13. Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption", EUROCRYPT '10 , pp. 62–91, 2010.
14. Allison Lewko and Brent Waters, "Decentralizing Attribute-Based Encryption", Eurocrypt 2011, LNCS 6632, pp. 568-588, 2011.
15. Tatsuaki Okamoto and Katsuyuki Takashima, "Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption", CRYPTO 2010, LNCS 6223, pp. 191–208, 2010.
16. Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters, "Secure attribute-based systems", In ACM Conference on Computer and Communications Security, pp. 99–112, 2006.
17. Amit Sahai and Brent Waters, "Fuzzy identity-based encryption", EUROCRYPT '05, pp. 457–473, 2005.
18. V. Shoup, "Lower bounds for discrete logarithms and related problems", EUROCRYPT 97, pages 256–266, 1997.
19. Patrick Traynor, Kevin R. B. Butler, William Enck, and Patrick McDaniel, "Realizing massive-scale conditional access systems through attribute-based cryptosystems", In NDSS, 2008.
20. Brent Waters, "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization", PKC 2011, LNCS 6571, pp. 53–70, 2011.