Attribute-Based Key-Insulated Encryption

JIAN-HONG CHEN1,2, YONG-TAO WANG3 AND KE-FEI CHEN1,3

1Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, 200240 P.R. China

2School of Computer Engineering
Huaiyin Institute of Technology
Huaian, 223003 P.R. China

3National Laboratory of Modern Communications
Chengdu, 610041 P.R. China

Attribute-based encryption (ABE) is an exciting alternative to public-key encryption, as ABE develops encryption systems with high expressiveness, without the need for a public key infrastructure (PKI) that makes publicly available the mapping between identities (sets of attributes), public keys, and validity of the latter. Any setting, PKI or attribute-based, must provide a means to revoke users from the system. To mitigate the limitation of ABE with regard to revocation, we propose an attribute-based key-insulated encryption (ABKIE) scheme, which is a novel ABE scheme. In our ABKIE scheme, a private key can be renewed without having to make changes to its public key (a set of attributes). The scheme is secure against adaptive chosen ciphertext attacks. The formal proof of security is presented under the Selective-ID security model, i.e., without random oracles, assuming the decision Bilinear Diffie-Hellman problem is computationally hard. To the best of our knowledge, this is the first ABKIE scheme up to now. Further, this is also the first concrete ABE construction with regard to revocation.

Keywords: attribute based, encryption, key insulation, selective-ID security model, revocation

1. INTRODUCTION

Attribute-based encryption (ABE) [1] is an exciting alternative to public-key encryption, which develops encryption systems with high expressiveness, without the need for a public key infrastructure (PKI) that makes publicly available the mapping between identities (sets of attributes), public keys, and validity of the latter. The ABE primitive provides some sort of error-tolerance, i.e., identities are viewed as sets of attributes, and a user can decrypt if it holds private keys for enough of (but not necessarily all) attributes a ciphertext is encrypted under. At the same time, colluding users cannot combine their keys to decrypt a ciphertext which none of them were able to decrypt independently.

Any setting, PKI or Attribute-based, must provide a means to revoke users from the system, e.g., if their private keys get compromised. Revocation was first studied in the attribute-based setting by Pirretti et al. in [2], which was refined by Boldyreva et al. [3]. In

Received June 22, 2009; revised September 15 & November 16, 2009 & February 1, 2010; accepted March 2, 2010.
Communicated by Chi-Jen Lu.

* This work was supported by the National Natural Science Foundation of China (No. 60970111), the National High Technology Research and Development Program (863) of China (No. 2009AA01Z418), the National Basic Research Program (973) of China (No. 2007CB311201) and the Foundation of NLMC (9140C1103020 803).
the methods of [2, 3], the authority updates private keys corresponding to identities (sets of attributes) periodically. Unfortunately, a concrete revocable-ABE construction wasn’t given in [2, 3]. Further, in the settings with a large number of users and short (e.g. per day) renewal interval, communication and computation overhead will make the authority insufferable due to frequent interaction with the authority. In 2002, Dodis et al. [4] introduced a key insulation mechanism, which can protect private keys in public key cryptosystems. The first key-insulated encryption scheme in [4] was improved by Bellare et al. [5]. Then, a parallel key-insulation mechanism [6] and a threshold key-insulation mechanism [7] were given for some special situations. In identity-based scenarios, Hanaoka et al. [8] put forward an identity-based key-insulated encryption (IBKIE) scheme. [7, 9] extended the threshold insulation and the parallel insulation respectively to identity-based settings.

To mitigate the limitation of ABE with regard to revocation, we propose an attribute-based key-insulated encryption (ABKIE) scheme. In our ABKIE scheme, a user can update his private key without the help of the authority and without changing his identity (a set of attributes). We want to remove the authority from the process of key update and significantly minimize the work done by the authority. First we define the ABKIE primitive and its security model that formalizes the possible threats. The model, of course, takes into account all adversarial capabilities of the standard ABE security notion. Then, based on the new model, we give a concrete construction. To the best of our knowledge, this is the first ABKIE scheme up to now. Further, this is also the first concrete ABE construction with regard to revocation. ABKIE can be considered a generalization of IBKIE. Our ABKIE (secure under the Selective-ID security model, i.e. without random oracles) is more secure and more expressive than Hanaoka et al.’s [8] IBKIE (secure in the random oracle model). High expressiveness is the merit we inherit from ABE. We assume the lifetime of the system is divided into $N$ time periods and $k$ denotes the number of time periods that may be compromised by the adversary. Dodis et al. [4] gave a $(k, N)$-key-insulated encryption scheme. Their scheme, however scaled poorly, having cost proportional to $k$. Bellare et al. [5] refined it and put forward a strongly key-insulated encryption scheme with optimal threshold. In Bellare et al.’s scheme, $k$ need not be known in advance and can be as large as one less than the total number of periods, yet the cost of the scheme is not impacted. We follow the Bellare et al.’s work. Thus, our ABKIE scheme is also with optimal threshold. Our proposed scheme is $(N - 1, N)$-key-insulated, i.e. even if temporary private keys for up to $N - 1$ time periods are compromised, an adversary is still unable to derive this user’s temporary private key from the remaining time period. Further, it is strongly key-insulated, i.e. even if the adversary exposes the secrets stored in the helper, there is still no security compromise.

The rest of this paper is organized as follows: In section 2, we provide some preliminaries for an ABKIE scheme. Section 3 gives the syntax definition and security notions of the scheme. Then in section 4, we detail our ABKIE construction and the security proof. Section 5 is our conclusion.

2. PRELIMINARIES

Throughout this paper, we let $\mathbb{Z}_p$ denote the set $\{0, 1, 2, \ldots, p - 1\}$ and $\mathbb{Z}_p'$ denote $\mathbb{Z}_p/0$. For a finite set $S$, $x \in_S S$ mean choosing an element $x$ from $S$ with a uniform distribution.
2.1 Bilinear Pairings

Let \( G_1 \) and \( G_2 \) be two cyclic multiplicative groups with the same prime order \( p \). Let \( e: G_1 \times G_1 \rightarrow G_2 \) be a pairing which satisfies the following conditions:

- Bilinear: For all \( g_1, g_2 \in G_1 \) and for all \( a, b \in \mathbb{Z}_p^* \), we have \( e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} \).
- Non-degenerate: There exists \( g_1, g_2 \in G_1 \) such that \( e(g_1, g_2) \neq 1 \).
- Computable: There is an efficient algorithm to compute \( e(g_1, g_2) \) for all \( g_1, g_2 \in G_1 \).

2.2 The DBDH Assumption

**Definition 1** The Decisional Bilinear Diffie-Hellman (DBDH) Assumption [1, 10] is that no probabilistic polynomial-time algorithm \( B \) that outputs \( b \in \{0, 1\} \) can distinguish the tuple \( (A = g^a, B = g^b, C = g^c, e(g, g)^{abc}) \) from the tuple \( (g, A = g^a, B = g^b, C = g^c, e(g, g)^z) \), where \( a, b, c, z \in \mathbb{Z}_p \). The advantage of \( B \) is \( |\Pr[B(g, A, B, C, e(g, g)^z) = 0] - \Pr[B(g, A, B, C, e(g, g)^z) = 1]| \).

2.3 PRF

Before describing the application of Pseudorandom Functions (PRFs) in our concrete ABKIE scheme, let us recall the definition [11] of the pseudorandom collection from which PRFs are picked.

**Definition 2** Let \( I_k \) denote the set of all \( k \)-bit strings and \( H_k \) denote the set of all functions from \( I_k \) to \( I_k \). We restrict ourselves to choose functions from a subset \( F_k \in H_k \) where the pseudorandom collection \( F = \{F_k\} \) has the following properties:

- Indexing: Each function in \( F_k \) has a unique \( k \)-bit index associated with it. (Thus picking randomly a function \( f \in F_k \) is easy.)
- Poly-time Evaluation: There exists a polynomial time Turing machine that given an index of a function \( f \in F_k \) and an input \( x \), computes \( f(x) \).
- Pseudorandomness: No probabilistic algorithm that runs in time polynomial in \( k \) can distinguish the functions in \( F_k \) from the functions in \( H_k \).

### 3. MODEL OF ABKIE

An ABKIE scheme consists of six algorithms:

- **Setup\( (d) \)**: Given a threshold value \( d \), the authority runs this algorithm to output a master key \( MK \) and a set of public parameters \( PK \).
- **KeyGen\( (\omega, MK) \)**: Given the user’s identity \( \omega \), as a set representing a user’s attributes, and the master-key \( MK \), the authority runs this algorithm to output an initial private key \( TK_{\omega,0} \) and a helper key \( HK_{\omega} \) corresponding to \( \omega \). The helper key is kept by the helper and the user with identity \( \omega \) keeps the initial private key.
- **HelperUpt\( (t, t', \omega, HK_{\omega}, PK) \)**: Given period indices \( t \) and \( t' \), an identity \( \omega \), its helper key
and the public parameters $PK$, the helper runs this algorithm to output the key-update information $UI_{\omega,t}$ for $\omega$ from period $t'$ to period $t$.

- **UserUpt($t, t', \omega, TK_{\omega,t'}, UI_{\omega,t'}, PK$):** Given period indices $t$ and $t'$, an identity $\omega$, the temporary private key $TK_{\omega,t'}$ corresponding to $\omega$ and $t'$, the key-update information $UI_{\omega,t'}$, for $\omega$ from period $t'$ to period $t$ and the public parameters $PK$, the user with identity $\omega$ runs this algorithm to output the temporary private key $TK_{\omega,t}$ corresponding to $\omega$ and $t$.

- **Encryption($t, M, \omega', PK$):** The Encryption algorithm is run by a user to encrypt a message $M$, with a target identity $\omega'$, period $t$ and the public parameters $PK$. It outputs a ciphertext, $E$ encrypted under $\omega'$ and $t$.

- **Decryption($E, \omega', \omega, TK_{\omega,t}, PK$):** The Decryption algorithm is run by a user with identity $\omega$ and the temporary private key $TK_{\omega,t}$ to attempt to decrypt a ciphertext $E$ under identity $\omega'$ and period $t$. If the set overlap $|\omega \cap \omega'| \geq d$, the algorithm will output the decrypted message $M$.

The security notions for ABKIE schemes are based on the security definitions in key-insulated encryption [4, 5] and ABE systems [1].

### 3.1 Key-insulated Security

We first consider the basic (i.e., non-strongly) key-insulated security for ABKIE. For one thing, as standard ABE systems, the key generation queries should be considered. For another, as traditional key-insulated encryption schemes, the temporary private key exposure should be addressed. An ABKIE scheme is said to be secure against chosen plaintext attacks (CPA) in the sense of key-insulation if no probabilistic polynomial-time adversaries have non-negligible advantage in the following game.

**Init** The adversary declares the identity $\gamma^*$ and the time period index $t^*$ that he wishes to be challenged upon.

**Setup** The challenger runs the setup phase of the algorithm and tells the adversary the public parameters.

**Phase 1** The adversary adaptively issues a set of queries as below:

- **Key Generation Query ($\gamma$):** The challenger first runs algorithm KeyGen to obtain the initial private key $TK_{\gamma,0}$ and the helper key $HK_{\gamma}$ corresponding to identity $\gamma$. It then sends these results to the adversary.

- **Temporary Private Key Query ($\gamma, t$):** The challenger runs algorithm UserUpt to obtain the temporary private key for identity $\gamma$ and period index $t$. It then sends this result to the adversary.

**Challenge** The adversary submits two equal length messages $M_0, M_1$. The challenger flips a random coin, $b$, and encrypts $M_b$ with $\gamma^*$ and $t^*$. The ciphertext is passed to the adversary.

**Phase 2** Phase 1 is repeated.
Guess The adversary outputs a guess $b'$ of $b$.

For convenience, we give the definition of a restricted identity as below: the set overlap between a restricted identity and the challenge identity $\gamma^*$ is at least $d$. The advantage of an adversary $A$ in this game is defined as $|\Pr[b' = b] - \frac{1}{2}|$. We refer to the above game as an IND-A&KI-CPA game. In the above game, it is mandated that the following conditions are simultaneously satisfied: (1) $A$ is disallowed to issue key generation queries for the restricted identities; (2) $A$ is disallowed to issue temporary private key queries for the restricted identities and the challenged time period $t^*$.

**Remark 1**: For those non-restricted identities, Temporary Private Key Query is of no help for $A$, since he can derive the temporary private key for any of these identities by issuing key generation queries. Thus, without loss of generality, we require that $A$ should only issue temporary private key queries for the restricted identities.

### 3.2 Strongly Key-insulated Security

The strongly key-insulated security for key-insulated encryption systems means that, if an adversary does not compromise any private key, exposure of the helper key does not enable the adversary to decrypt a valid ciphertext for any time period. The word “strongly” doesn’t mean our strongly key-insulated security is more strong security than our key-insulated security. We inherit the notion of strongly key-insulated security from [4, 5] and develop it in the attribute based scenario. To model this security notion for ABKIE systems, we allow the adversary to compromise the helper key. An ABKIE scheme is said to be secure against chosen plaintext attacks (CPA) in the sense of strong key-insulation if no probabilistic polynomial-time adversaries have non-negligible advantage in an IND-A&SKI-CPA game. The IND-A&SKI-CPA game is almost the same as the IND-A&KI-CPA game except Phase 1.

**Phase 1** The adversary adaptively issues a set of queries as below:

- Key Generation Query $\langle \gamma \rangle$: the same as the IND-A&KI-CPA game.
- Helper Key Query $\langle \gamma \rangle$: The challenger runs algorithm KeyGen to generate $HK$, and sends it to the adversary.

The advantage of an adversary $A$ in this game is defined as $|\Pr[b' = b] - \frac{1}{2}|$. In the above game, it is mandated that the following condition is satisfied: $A$ is disallowed to issue key generation queries for the restricted identities.

**Remark 2**: For those non-restricted identities, Helper Key Query is of no help for $A$, because he can derive the helper key for any of these identities by issuing key generation queries. Thus, without loss of generality, we require that $A$ should only issue helper key queries for the restricted identities.
4. OUR PROPOSED ABKIE SCHEME

4.1 Description of Our Scheme

Our proposed ABKIE scheme is based on Sahai-Waters’ large universe ABE construction [1]. We first give a brief review of Sahai-Waters’ construction to show the difference between it and our ABKIE construction. Let \( G_1 \) and \( G_2 \) be two groups with prime order \( p \) of size \( \kappa \) and let \( g \) be a generator of \( G_1 \). Additionally, let \( e: G_1 \times G_1 \rightarrow G_2 \) denote the bilinear map. We restrict encryption identities to be of length \( n \) for some fixed \( n \).

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 \( \mathbb{Z}_p \).

Identities will be sets of \( n \) elements of \( \mathbb{Z}_p \). With some minor modifications to our scheme, which we omit for simplicity, we can encrypt to all identities of size \( \leq n \). Note that here we associate each element with a unique integer in \( \mathbb{Z}_p \), while in practice an attribute will be associated with each element so that identities will have some semantics. We can also describe an identity as a collection of \( n \) strings of arbitrary length and hash strings into members of \( \mathbb{Z}_p \) using a collision-resistant hash function, \( H: \{0, 1\}^* \rightarrow \mathbb{Z}_p \).

Sahai-Waters’ ABE scheme consists of the following algorithms:

- Setup: The authority picks \( y \in \mathbb{Z}_p \), \( g_2 \in \mathbb{G} \), sets \( g_1 = g^y \), picks \( v_1, \ldots, v_{n+1} \in \mathbb{G} \), lets \( N = \{1, \ldots, n+1\} \) and defines a function, \( V \), as \( V(x) = g_2^{\sum_{i=1}^{n+1} \lambda_{i,S}(x)} \). We can view \( V \) as the function \( g_2^{\sum_{i=1}^{n+1} b_i(x)} \) for some \( n \) degree polynomial \( b \). The public parameters are published as \( PK = (g_1^y, g_2, v_1, \ldots, v_{n+1}) \) and the master secret key is \( MK = y \).

- KeyGen: To generate the private key for identity \( \omega \), the authority first chooses a \( d - 1 \) degree polynomial \( q(x) \) randomly such that \( q(0) = y \). Then, for all \( i \in \omega \), it picks \( r_i \in \mathbb{Z}_p \) and computes the private key \( SK_\omega = (\{d_{i,c} | c \in \omega\} = (\{g_2^r V(i)^{y}\} | c \in \omega\} g_2^{y}) \).

- Encryption: To encrypt a message \( M \in \mathbb{G} \) with the public key \( \alpha \), a user picks \( s \in \mathbb{Z}_p \) and publishes the ciphertext as \( E = (\alpha, E' = M g_1^{g_2^{y}} V(i)^{y}, E'' = g_2^{y}, \{E_i = V(i)^{y}\} | i \in \omega \) \).

- Decryption: Suppose that a ciphertext, \( E \), is encrypted with identity \( \alpha \) and we have a private key for identity \( \omega \), which we refer to as \( \omega \cap \alpha \). The decryption algorithm interpolates a polynomial in the exponent using Shamir’s [12] secret sharing method. For each \( i \in \omega \), the user with identity \( \omega \) and private key \( SK_\omega \) computes a temporary value \( A_i = e(D_i, E') = \frac{e(g_2^{y}, V(i)^{y}) e(g_2^{y} V(i)^{y})}{e(g_2^{y} V(i)^{y})} = e(g_2^{y}, V(i)^{y}) \). Using polynomial interpolation the algorithm recovers the value \( e(g_2^{y}) \) and divides it out by computing:

\[
\prod_{i \in \omega} A_i^{\lambda_{i,S}(0)} = \prod_{i \in \omega} e(g_2^{y} V(i)^{y})^{\lambda_{i,S}(0)} = \frac{E'}{e(g_2^{y})^{\lambda_{i,S}(0)}} = \frac{E'}{e(g_2^{y})} = M.
\]
rinary private key update is even more difficult in attribute systems, given that each attribute is conceivably possessed by multiple different users, whereas public/private key pairs are uniquely associated with a single user. Hence, constructing an ABKIE scheme by extending IBKIE isn’t so straightforward. We base our concrete scheme on Sahai-Waters ABE construction. We notice that adding key insulation mechanism to ABE construction is not trivial. In Sahai-Waters construction, it is required by provable security that exponents in temporary private keys should be random numbers. On the other hand, the randomness of exponents in temporary private keys makes the user of the current time period unable to derive the random exponents of the time periods before. Consequently, it is very difficult to implement temporary private key update algorithm. To solve this problem, inspired by the cryptographic applications of pseudo-random functions (PRFs) in [13], we use a PRF family $F$ such that given a $κ$-bit seed (index) $s$ and a $κ$-bit argument (input) $x$, it outputs a $κ$-bit string $F_s(x)$. In our concrete ABKIE scheme, identities will be sets of $n$ elements of $Z_p$ and period indices will be elements of $Z$. The proposed ABKIE scheme consists of the following algorithms:

- Setup: The authority picks $y ∈ R Z_p$, $g_2 ∈ G_1$, $h_1 ∈ R G_1$, sets $g_1 = g^{y}$, picks $v_1, ..., v_{n+1} ∈ R G_1$, lets $N$ be the set $\{1, ..., n + 1\}$ and defines a function, $V$, as $V(x) = g_2^{x} \prod_{i=1}^{n+1} v_i^{\lambda_i(x)}$. We can view $V$ as the function $g_2^{x^h}$ for some $n$ degree polynomial $h$. For clarity, we define $H_{\omega}: Z_p → G_1$ to be the function $H_{\omega}(x) = g_1^x h_1$. The public parameters are published as $PK = (g_1, g_2, h_1, v_1, ..., v_{n+1})$ and the master secret key is $MK = y$.
- KeyGen: To generate the helper key and the initial private key for identity $\omega$, the authority first picks a helper key $HK_{\omega} ∈ R \{0, 1\}^*$, computes $k_{0,\omega} = F_{HK_{\omega}}(0)$ and chooses a $d - 1$ degree polynomial $q(x)$ randomly such that $q(0) = y$. Note that if the length of the input for $F$ is less than $κ$, we can add some “0”s as the prefix to meet the length requirement. Then, for all $i ∈ \omega$, it picks $r_i ∈ R Z_p$ and computes the initial private key $TK_{\omega} = ((D_1^{\omega}_1)_{i ∈ \omega}, D_2^{\omega}_1, (d_i)_{i ∈ \omega}) = ((g_2^0)^V(i)H_{\omega}(0)^{x}\omega)_i ∈ \omega, g^{k_{0,\omega}}, (g^{x}_i)_{i ∈ \omega})$. The helper key is kept by the helper and the user with identity $\omega$ keeps the initial private key.
- HelperUpt: This algorithm first computes $k_{0,\omega} = F_{HK_{\omega}}(t)$ and $k_{0,\omega} = F_{HK_{\omega}}(t')$. Then it defines and returns the key-update information for identity $\omega$ from period $t$ to period $t$ as $UI_{\omega, t} = (U^{D_1^{\omega}_1}_{\omega, t}, U^{D_2^{\omega}_1}_{\omega, t}) = (H_{\omega}(t)\omega_1^\omega)\omega_2^\omega, g^{k_{0,\omega}}$.
- UserUpt: This algorithm first parses the temporary private key for identity $\omega$ and period $t'$ as $TK_{\omega} = ((D_1^{\omega}_1)_{i ∈ \omega}, D_2^{\omega}_1, (d_i)_{i ∈ \omega})$ and parses the key-update information for identity $\omega$ from period $t$ to period $t$ as $UI_{\omega, t} = (U^{D_1^{\omega}_1}_{\omega, t}, U^{D_2^{\omega}_1}_{\omega, t})$. Then it sets the temporary private key for identity $\omega$ and period $t$ as $TK_{\omega} = ((D_1^{\omega}_1)_{i ∈ \omega}, D_2^{\omega}_1, (d_i)_{i ∈ \omega}) = ((D_1^{\omega}_1), U^{D_1^{\omega}_1}_{\omega, t}, U^{D_2^{\omega}_1}_{\omega, t}, (d_i)_{i ∈ \omega})$, deletes $TK_{\omega}$ and $UI_{\omega, t}$, and returns $TK_{\omega}$. Note that in time period $t$, $TK_{\omega}$ is always set to be $((D_1^{\omega}_1)_{i ∈ \omega}, D_2^{\omega}_1, (d_i)_{i ∈ \omega}) = ((g_2^0)^V(i)H_{\omega}(t)^{x}\omega)_i ∈ \omega, g^{k_{0,\omega}}, (g^{x}_i)_{i ∈ \omega})$.
- Encryption: In time period $t$, to encrypt a message $M ∈ G_1$ with the public key $\omega'$, a user picks $s ∈ R Z_p$ and publishes the ciphertext as $E = (t, \omega, E^t = Me(g_1, g_2^y, E^{\omega'} = g_1^s, E^{t'} = H_{\omega}(0), E_t = V(I_1^{\omega'}))$.
- Decryption: Suppose that a ciphertext, $E$, is encrypted with identity $\omega'$ and period index $t$ while we have a temporary private key for identity $\omega$ and period index $t$, where $|\omega' \cap \omega| ≥ d$. Choose an arbitrary $d$-element subset, $S$, of $\omega' \cap \omega$. Then, the ciphertext can be decrypted as
Theorem 1 Our ABKIE scheme is IND-A&KI-CPA secure in the Selective-ID model, assuming that the DBDH assumption holds in groups \((G_1, G_2)\) and \(F\) is a family of PRFs. Concretely, if there exists an \(\text{IND-A&KI-CPA}\) adversary \(A\) against our scheme, then a simulator \(B\) can be constructed to distinguishes a DBDH tuple from a random tuple.

Proof: Suppose a polynomial-time adversary \(A\) can win the \(\text{IND-A&KI-CPA}\) game with advantage \(\epsilon\). We build a simulator \(B\) that can distinguish a DBDH tuple from a random tuple with advantage \(\frac{\epsilon}{2}\). The challenger first sets the groups \(G_1\) and \(G_2\) with an efficient bilinear map \(e\) and a generator \(g\). Then, the challenger picks \(a, b, c, z \in \mathbb{Z}_p\), \(\mu \in \{0, 1\}\). If \(\mu = 0\), the DBDH challenger sets \((A, B, C, Z) = (g^a, g^b, g^c, e(g, g)^{\mu})\); otherwise it sets \((A, B, C, Z) = (g^a, g^b, g^c, e(g, g)^{\mu})\). Let \(g_1 = A\) and \(g_2 = B\). The challenger then gives \((g, A, B, C, Z)\) to \(B\). \(B\) now plays the role of challenger in the \(\text{IND-A&KI-CPA}\) game.

\text{Init} \quad \text{During this phase, \(B\) receives the challenge identity \(\gamma^*\) (an \(n\) element set of members of \(Z_p\)) and the challenge period index \(i\).

\text{Setup} \quad \text{\(B\) chooses a random \(n\) degree polynomial \(f(x)\) and calculates an \(n\) degree polynomial \(u(x)\) such that \(u(x) = -x^d\) for all \(x \in \gamma^*\) and where \(u(x) \neq -x^d\) for some other \(x\). Since \(-x^d\) and \(u(x)\) are two \(n\) degree polynomials they will either agree on at most \(n\) points or they are the same polynomial. Our construction assures that \(\forall x \ u(x) = -x^d\) if and only if \(x \in \gamma^*\). Then, for \(i\) from 1 to \(n + 1\) \(B\) sets \(v_i = g_2^{x\cdot u(x)}\). Note that since \(f(x)\) is a random \(n\) degree polynomial \(v_i\) will be chosen independently at random as in the construction and we implicitly have \(V(x) = g_2^{x\cdot f(x)}\). In addition, \(B\) picks \(\beta \in \mathbb{Z}_p\) and defines \(h_1 = g_1^{\beta} g_2^d\). Finally \(B\) gives \(A\) the public parameters, \(PK = (g_1, g_2, v_1, \ldots, v_{n+1})\). Observe that from the perspective of \(A\), the distributions of these public parameters are identical to the real construction. As before, we define \(H_u: \mathbb{Z} \rightarrow G_1\) to be the function

\[ H_u(x) = g_1^{\beta} h_1 = g_1^{\beta + x^d}. \]

Phase 1 \quad \text{\(A\) issues a series of queries as in the definition of the \(\text{IND-A&KI-CPA}\) game.}

- \text{Key Generation Queries:} \(B\) maintains a list \(HK_{\text{init}}\) which is initially empty. Suppose \(A\) requests a helper key and an initial private key for identity \(\gamma\). \(B\) first checks whether \(HK_{\text{init}}\) has contained a tuple for this input. If yes, the predefined value is returned to \(A\).
Otherwise, it picks $HK_y \in \mathbb{R} \{0, 1\}^k$. Next, it adds tuple $(\gamma, HK_y)$ to list $HK^{\text{int}}$, and returns $HK_y$ to $A$. We set $\Gamma = \gamma \cap \gamma'$, let $\Gamma'$ be any set such that $\Gamma \subseteq \Gamma' \subseteq \gamma'$ and $|\Gamma'| = d - 1$, and set $S = \Gamma' \cup \{0\}$. $B$ computes $k_{0,0} = F_{HK_y}(0)$, sets $D_{\gamma,0}^{(0)} = g^{\beta^2} g^{\gamma_{b,0}}$ and sets $\tilde{k}_{\gamma,0} = k_{\gamma,0} = \frac{\beta}{b - \Gamma}$. Then, we have $D_{\gamma,0}^{(0)} = g^{\tilde{k}_{\gamma,0}}$ and

$$g^{\beta^2} H_{w}(0)^{k_{b,0}} = g^{\beta^2} (g_1^{0-i} g^\gamma)^{k_{b,0}} g^{ab} (g_1^{0-i} g^\gamma)^{k_{b,0}} g^{ab} (g_1^{0-i} g^\gamma)^{k_{b,0}} = g^{\beta^2} (g_1^{0-i} g^\gamma)^{k_{b,0}} (g_1^{0-i} g^\gamma)^{k_{b,0}} (g_1^{0-i} g^\gamma)^{k_{b,0}} \frac{b}{b - \Gamma}$$

(1)

For $i \in \Gamma'$, $B$ picks $\tilde{\tau}_i, \lambda_i \in \mathbb{R} Z_p$, sets

$$D_{\gamma,0}^{(1)} = g^{\beta^2} V(i)^{\gamma} g^{0-i} H_{w}(0)^{k_{b,0}} = g^{\beta^2} V(i)^{\gamma} g^{\beta^2} H_{w}(0)^{k_{b,0}}$$

(2)

For $i \in \gamma - \Gamma'$, $B$ sets

$$D_{\gamma,0}^{(1)} = (\prod_{j \in \Gamma'} g_2^{2 \lambda_a j(i)}) (g_1^{2 \lambda_a u(i)} (g_2^{2 \lambda_a u(i)} g^{f(i)})^{\lambda_{a,i}(0)}) \frac{\beta^2}{g_2^{0-i} H_{w}(0)^{k_{b,0}}}$$

(3)

$$d_i = (g_1^{\lambda_{a,i}(i)} g^{k_{b,0}})^{\lambda_{a,i}(i)}$$

The value $\mu + u(i)$ will be non-zero for all $i \in \gamma'$, which includes all $i \in \gamma - \Gamma'$. This follows from our construction of $\alpha(i)$. Observe that if let $r_i = (\tilde{\tau}_i - \frac{\beta}{\mu + u(i)}) \Delta_{a,b}(i)$, the initial private key components in Eq. (3) have the same form as those in Eq. (1). To see this, let $q(x)$ denote the $(d - 1)$-degree polynomial such that $q(0) = a$ and $q(i) = \lambda_i$ for each $i \in \Gamma'$. Besides, for each $i \in \gamma - \Gamma'$, we let $q(i)$ denote $\lambda_i$. Then, for each $i \in \gamma - \Gamma'$, we have

$$D_{\gamma,0}^{(1)} = (\prod_{j \in \Gamma'} g_2^{2 \lambda_a j(i)}) (g_1^{2 \lambda_a u(i)} (g_2^{2 \lambda_a u(i)} g^{f(i)})^{\lambda_{a,i}(0)}) \frac{\beta^2}{g_2^{0-i} H_{w}(0)^{k_{b,0}}}$$

(4)
\[ = (\prod_{j \in \Gamma'} g_2^{x_j t'(i)}) (g_2^{x_0 + u(i)}) \left( g_2^{\delta + u(i)} \right) \left( g_2^{\delta - u(i)} \right) \left( g_2^{\delta + u(i)} \right) g_2^a H_u(0)^{\delta, t'} \]
\[ = g_2^{\delta} V(i)^{\delta} g_2^{\delta, t'} H_u(0)^{\delta, t'} = g_2^{\delta} V(i)^{\delta} g_2^a H_u(0)^{\delta, t'} \]
\[ d_i = (g_1^{\alpha} g_2^{\delta}) \langle \rangle = (g_1^{\alpha} g_2^{\delta}) \langle \rangle = g_1^{\alpha} \cdot g_2^{\delta} \]

Temporary Private Key Queries: Suppose \( A \) requests a temporary private key for identity \( \gamma \) and time period \( t \). We define \( \Gamma, \Gamma' \) and \( S \) in the same way as Key Generation Queries. \( B \) picks \( k_{t, u} \in \mathbb{Z}_p \). Note that since \( F \) is a PRF family and \( A \) does not know the corresponding seed \( \text{HK}_n \), the exponents \( k_{t, u} \) is indistinguishable from the real construction in \( A \)'s view and \( B \) can freely define \( k_{t, u} \) himself. \( B \) sets \( D_{t, i}^L = g_2^{\delta} \) and sets \( k_{t, u} = k_{t, u} - \frac{b}{l} \). Then, we have \( D_{t, i}^L = g_2^{\delta} \). Similarly with Eq. (1), we have \( g_2^{\delta} H_u(t)^{\delta, t} = g_2^{\delta} H_u(t)^{\delta, t} \). For \( i \in \Gamma \), \( B \) picks \( t_k, p \in \mathbb{Z}_p \) and sets
\[ D_{t, i}^L = g_2^{\delta} V(i)^{\delta} g_2^{\delta, t'} H_u(t)^{\delta, t'} = g_2^{\delta} V(i)^{\delta} g_2^a H_u(t)^{\delta, t'}, d_i = g_1^{\alpha}. \] (5)

For \( i \in \gamma - \Gamma \), \( B \) sets
\[ D_{t, i}^L = \prod_{j \in \Gamma'} g_2^{x_j t'(i)} (g_1^{\alpha} g_2^{\delta}) \langle \rangle = g_2^{\delta} V(i)^{\delta} g_2^{\delta, t'} H_u(t)^{\delta, t'}, \]
\[ d_i = (g_1^{\alpha} g_2^{\delta}) \langle \rangle = (g_1^{\alpha} g_2^{\delta}) \langle \rangle = g_1^{\alpha} \cdot g_2^{\delta} \] (6)

Similarly with Eq. (4), the temporary private key components in Eq. (5) have the same form as those in Eq. (6).

Challenge \( A \) will submit two challenge messages \( M_1 \) and \( M_0 \) to the simulator. The simulator flips a fair binary coin, \( v \), and returns an encryption of \( M_v \). The ciphertext is \( E = \left( i', \gamma', E' = M, E'' = C, E'' = C' \right) \). If \( \mu = 0 \), then \( Z = e(g, g)^{abc} \). The ciphertext is \( E = \left( i', \gamma', E' = M, e(g, g)^{abc}, E'' = C' \right) \). If \( \mu = 1 \), then \( Z = e(g, g)^{abc} \). Since \( z \) is random, \( E' \) will be a random element of \( G_2 \) from the adversaries’ view and the message contains no information about \( M_v \).

Phase 2 The simulator acts exactly as it did in Phase 1.

Guess \( A \) will produce a guess \( v' \) of \( v \). If \( v' = v \), \( B \) answers “DBDH” in the DBDH game to indicate that it was given a BDH-tuple. Otherwise, \( B \) answers “random” to indicate it was given a random 4-tuple.

If \( Z = e(g, g)^{abc} \), \( E \) is a valid ciphertext, in which case the advantage of \( A \) is \( e \). As a
result, \( \Pr[B \rightarrow \text{"DBDH"} | \mu = 0] = \Pr[B \rightarrow \text{"DBDH"} | Z = e(g, g)^{abc}] = \Pr[\nu' = v | Z = e(g, g)^{abc}] \). Since \( B \) guesses \( \mu' = 0 \) when \( \nu' = v \), we have \( \Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \epsilon \). If \( Z = e(g, g)^{abc} \), then \( E' \) is completely random from the view of \( A \). Hence, \( \Pr[B \rightarrow \text{"random"} | \mu = 1] = \Pr[B \rightarrow \text{"random"} | Z = e(g, g)^{abc}] = \Pr[\nu' \neq v | Z = e(g, g)^{abc}] = \frac{1}{2} \). Since \( B \) guesses \( \mu' = 1 \) when \( \nu' \neq v \), we have \( \Pr[\mu' = \mu | \mu = 1] = \frac{1}{2} \). The overall advantage of \( B \) in the DBDH game is

\[
\frac{1}{2} \Pr[\mu' = \mu | \mu = 0] + \frac{1}{2} \Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2} \left( \frac{1}{2} + \epsilon \right) + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} \cdot \epsilon
\]

\[ \Box \]

**Theorem 2** Our ABKIE scheme is IND-A&SKI-CPA secure in the Selective-ID model, assuming that the DBDH assumption holds in groups \((G_1, G_2)\) and \(F\) is a family of PRFs. Concretely, if there exists an IND-A&SKI-CPA adversary \( A \) against our scheme, then a simulator \( B \) can be constructed to distinguishes a DBDH tuple from a random tuple.

**Proof:** The proof of Theorem 2 is almost the same as Theorem 1 except Phase 1.

**Phase 1** \( A \) issues a series of queries as in the definition of the IND-A&SKI-CPA game.

– Helper Key Queries: \( B \) maintains a list \( HK_{\text{list}} \) which is initially empty. On receiving a helper key query \( \langle \gamma \rangle \), \( B \) first checks whether \( HK_{\text{list}} \) has contained a tuple for this input. If yes, the predefined value is returned to \( A \). Otherwise, it picks \( HK_{\gamma} \in \{0, 1\}^\kappa \). Next, it adds tuple \( \langle \gamma, HK_{\gamma} \rangle \) to list \( HK_{\text{list}} \) and returns \( HK_{\gamma} \) to \( A \).

– Key Generation Queries: The same as Theorem 1 except that \( B \) issues helper key queries on \( \langle \gamma \rangle \) to obtain \( HK_{\gamma} \) and computes \( k_{\gamma,0} = F_{HK_{\gamma}}(0) \).

\[ \Box \]

**4.4 Chosen-Ciphertext Security**

Similarly to [1], we can achieve the chosen-ciphertext security by applying the technique of using simulation-sound NIZK proofs to achieve chosen-ciphertext security [14]. We can also use other methods [15-17] to achieve greater efficiency.

**5. CONCLUSION**

We devised the first ABKIE scheme, a novel ABE scheme, to deal with the limitation of ABE with regard to revocation. The scheme is secure against adaptive chosen ciphertext attacks under the Selective-ID security model.

**ACKNOWLEDGMENTS**

We thank anonymous reviewers for their invaluable comments.

**REFERENCES**

1. A. Sahai and B. Waters, “Fuzzy identity-based encryption,” in *Proceedings of EURO-
Jian-Hong Chen (陳劍洪) received his M.S. and B.S. degrees in Computer Science and Engineering from University of Science and Technology Liaoning, Anshan, China, in 2001 and 1994, respectively. He is currently a Ph.D. candidate at Shanghai Jiao Tong University. His research interests include public key cryptosystem and pairing based cryptosystem.

Yong-Tao Wang (王永濤) received his M.S. degree in Computer Science and Engineering from Xihua University, Cheng-du, China, in 2007. He is currently a Ph.D. candidate at Shanghai Jiao Tong University. His research interests include public key cryptosystem and key agreement protocol.

Ke-Fei Chen (陳克非) received his Ph.D. degree from Justus Liebig University Giessen, Germany, in 1994. Since 1996, he came to Shanghai Jiao Tong University and became the Professor at the Department of Computer Science and Engineering. His areas of research include classical and modern cryptography, theory of network security, etc.