Attribute-Based Traitor Tracing*

YONG-TAO WANG1, KE-FEI CHEN1,2 AND JIAN-HONG CHEN1,3

1Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, 200240 P.R. China
2National Laboratory of Modern Communications
Chengdu, 610041 P.R. China
3Huaiyin Institute of Technology
Huaiian, 223003 P.R. China

In this paper, we focus on traitor tracing scheme in attribute-based encryption (ABE) scenarios. A well-known concern in the setting of attribute-based encryption is that a user (or set of colluding users) can create a new key (or decryption device) by using his legal one and distribute it for malicious use. To mitigate this problem, we introduce the notion of attribute-based traitor tracing (ABTT). We formalize the definitions and security notions for attribute-based traitor tracing scheme, and then present a construction of ABTT. Our scheme makes use of the identity-based traitor tracing technology proposed by Abdalla et al., and is based on the second construction of Sahai-Waters’ attribute-based encryption schemes, but in the asymmetric bilinear setting. Our scheme is shown to be secure in the standard model under some reasonable assumptions. To the best of our knowledge, this is the first ABTT scheme up to now.

Keywords: attribute-based encryption, traitor tracing, access rights, collusion, decryption device

1. INTRODUCTION

Attribute-based encryption, first introduced by Sahai and Waters [1] in 2005, is a new means for encrypted access control. In an attribute-based encryption system, the private keys of users are issued by a trusted authority, and both users’ private keys and ciphertexts will be associated with a set of attributes or a policy over attributes. A user is able to decrypt a ciphertext if there is a match between his private key and the ciphertext.

The original system of Sahai and Waters is a Threshold ABE system in which ciphertexts are labeled with a set of attributes $\omega$ and a user’s private key is associated with another set of attributes $\omega'$. Their schemes allow a user to decrypt a ciphertext when at least $d$ (threshold parameter) attributes overlapped between the ciphertext and the private key. At present, there are two approaches for ABE schemes to deploy access control policy. One is key-policy attribute-based encryption (KP-ABE) scheme first proposed by Goyal et al. [5]. In their cryptosystem, ciphertexts are labeled with sets of attributes, and private keys are identified by a tree-access structure in which each interior node of the tree is a threshold gate and the leaves are associated with attributes. This access tree control which ciphertexts a user is able to decrypt. The other approach to deploy policy is

Received March 24, 2009; accepted December 21, 2009.
Communicated by Wen-Guey Tzeng.

* This work was supported by the National Natural Science Foundation of China (No. 60970111), the National Basic Research Program (973) of China (No. 2007CB311201) and the Foundation of NLMC (9140C1103020803).
ciphertext-policy attribute-based encryption (CP-ABE) schemes [4, 14]. In these schemes, a message is encrypted with a specific access policy determined by the encrypter, and private keys issued by a trusted authority are labeled with sets of attributes. Decryption requires that the attribute set of a user matches the ciphertext’s policy. There are many applications for both types of ABE systems, such as sharing of audit-log and broadcast encryption. In addition, one can view ABE as a generalization of identity based encryption [2, 10, 11, 15, 19].

The motivation to consider traitor tracing in attribute-based encryption systems is relatively simple. In general, access rights are linked to users’ private keys in an ABE system. This leads to a problem that the system’s access control policy will be broken if a user (or set of colluding users) creates a key (or decryption device) and distributes it for malicious use [22]. However, in all known ABE schemes [1, 4-7, 13, 14], this problem is not addressed. In these schemes, users can easily give away their private keys (referred to as key cloning [22]) or just a subset of their private keys to create a decryption device (or key) which can decrypt ciphertexts or a subset of the ciphertexts. Collusion is easy for two or more users to construct new keys that can decrypt all (or some) of the ciphertexts, which those users can decrypt in ABE. The similar problem was solved by using a traitor tracing scheme in other settings in broadcast encryption [8, 9, 16, 18]. A traitor tracing scheme can trace any person (or set of colluding persons) who creates a new decryption device, or key. Thus considering above, we introduce the attribute-based traitor tracing scheme. We also notice that Hinek et al. introduced a new type of attribute-based encryption scheme [22], called token-based attribute-based encryption, which provides strong deterrence for key cloning, in the sense that delegation of keys reveals some personal information about the user. However, this work is orthogonal to ours.

In this paper, we consider the original ABE system proposed by Sahai and Waters. Our construction is based on the second construction of [1], but in the asymmetric bilinear setting. We take the tracing algorithm from identity-based traitor tracing [3], which proposed by Abdalla et al. at PKC 2007. We modify their algorithm and give the new proofs for our setting. Additionally, we notice that all methods used in the construction are general in some situations, i.e., by using these technologies we can construct an attribute-based traitor tracing based on other attribute-based encryption schemes. For example, key-policy ABE in [5].

Recall that ciphertexts are not encrypted to one particular user in an ABE system, in which both users’ private keys and ciphertexts will be associated with a set of attributes or a policy over attributes. This means that one ciphertext (implicitly) defines a group of users in which the user can decrypt this ciphertext, and that a user (implicitly) belongs to different groups defined by ciphertexts. Therefore, to classify users according to groups is a hard problem. In our construction, we use the collusion secure codes [17]. A user will be identified by a pair \((\omega, id)\), where \(\omega\) is a set of attributes and \(id\) denotes the user’s identity. It is allowable for different users with the same set of attributes. However, we require that a user should be assigned only one set of attributes. We will map \(id\) to a codeword. Notice that we need the identity \(id\) for a user be unique in our system. This is why our construction is not practical. We leave the construction of an efficient attribute-based traitor tracing scheme as an open problem. In addition, we only prove the traceability property of our traitor tracing system for the chosen-plaintext setting in a slightly weaker model known as the selective model. This is because the underlying ABE scheme is also proved
in the selective model. Thus, it is interesting to construct an ABTT system that satisfies the traceability property in the full model.

The rest of the paper is organized as follows. In section 2 we recall some preliminaries used by our scheme. In section 3 we formalize the definitions and security notions for attribute-based traitor tracing systems. We describe our attribute-based traitor tracing scheme in section 4 and prove the security of our construction in section 5. Finally, we conclude in section 6.

2. PRELIMINARIES

2.1 Bilinear Pairings and Complexity Assumptions

We now review the notion of asymmetric pairings [3]. Let \( G_1, G_2 \) and \( G_T \) denote three finite multiplicative abelian groups of prime order \( p \). Let \( g \) and \( h \) be generators of \( G_1 \) and \( G_2 \), respectively, and let \( \varphi: G_2 \rightarrow G_1 \) be an efficiently computable isomorphism such that \( \varphi(h) = g \). Assuming that there exists an admissible bilinear map \( \hat{e}: G_1 \times G_2 \rightarrow G_T \), which satisfy the following conditions:

- \( \hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab} \), for all \( a, b \in \mathbb{Z}_p \);
- \( \hat{e}(g^a, h^b) = 1 \) iff \( a = 0 \) or \( b = 0 \);
- \( \hat{e}(g^a, h^b) \) is efficiently computable.

We require the following assumptions hold which have been used before in [3, 12, 21].

**Definition 1** Computational Bilinear Diffie-Hellman (CBDH) Assumption in \( G_2 \): Suppose a challenger chooses \( a, b, c \in \mathbb{Z}_p \) at random. The CBDH assumption is that no polynomial-time adversary is to be able to compute \( \hat{e}(g, h)^{abc} \) from the tuple \( (A = h^a, B = h^b, C = h^c) \) with more than a negligible advantage.

**Definition 2** Decisional Bilinear Diffie-Hellman (DBDH) Assumption in \( G_2 \): Suppose a challenger chooses \( a, b, c, z \in \mathbb{Z}_p \) at random. The DBDH assumption is that no polynomial-time adversary is to be able to distinguish the tuple \( (A = h^a, B = h^b, C = h^c, Z = \hat{e}(g, h)^z) \) from the tuple \( (A = h^a, B = h^b, C = h^c, Z = \hat{e}(g, h)^y) \) with more than a negligible advantage.

**Definition 3** Decisional Diffie-Hellman (DDH) Assumption in \( G_1 \): Suppose a challenger chooses \( a, b, z \in \mathbb{Z}_p \) at random. The DDH assumption is that no polynomial-time adversary is to be able to distinguish the tuple \( (A = g^a, B = g^b, Z = g^z) \) from the tuple \( (A = g^a, B = g^b, Z = g^y) \) with more than a negligible advantage.

Note that assuming DDH problem is hard in \( G_1 \) means that there cannot exist a computable isomorphism from \( G_1 \) to \( G_2 \). To satisfy the above setting, we can choose some appropriate groups over MNT curves. For the details about MNT curves, we refer the reader to [20].
2.2 Collusion Secure Codes

This section recalls the notion of collusion secure codes [17]. The definitions follow from [3]. A \((c, N, \varepsilon)\)-collusion secure code, where \(N\) is the maximum number users in the system, \(c\) is the maximum number of colluders of a tracing algorithm can tolerate, and \(\varepsilon\) is the probability of error that a colluder is not traced, can be produced using codewords of size \(A = O(c^2 \log(N) + \log(1/\varepsilon))\) over an alphabet of size \(s = 2\).

Let \(\Sigma\) be a symbol alphabet of size \(|\Sigma| = s\). If \(x = x_1 \ldots x_A \in \Sigma^A\) is a string of \(A\) symbols and \(I = \{1 \leq i_1 < \ldots < i_n \leq A\}\) is a set of indices, then \(x|_I\) is the substring \(x_{i_1} \ldots x_{i_n}\) containing only those symbols of \(x\) at positions in \(I\). Let \(W = \{w_1, \ldots, w_c \in \Sigma^A\}\) be a set of symbol strings, and let \(I\) be the set of all positions where all strings in \(W\) are equal, i.e. \(I\) is the maximal set such that \(w_1|_I = w_2|_I = \ldots = w_c|_I\). Then the feasible set of \(W\) is defined as the set of all strings that are equal to \(w_1, \ldots, w_c\) at positions in \(I\), i.e. \(\text{FS}(W) = \{x \in \Sigma^A: x|_I = w_1|_I = w_2|_I = \ldots = w_c|_I\}\).

A \((c, N, \varepsilon)\)-collusion secure code of length \(A\) over alphabet \(\Sigma\) consists of a set \(C\), called the codebook, of indexed codewords \(w^r_i\) for \(1 \leq i \leq N\) and \(r \in \{0, 1\}^\rho\), and a tracing algorithm \(T_C\). These are such that for all collusions \(C \in \{1, \ldots, N\}\) of size at most \(c\), \(W = \{w^r_i: i \in C\}\), and for all (unbounded) algorithms \(A\) it holds that

\[
\Pr[T_C(x, r) \in C | x \in \text{FS}(W); x \xleftarrow{\$} A(W); r \xleftarrow{\$} \{0, 1\}^\rho] > 1 - \varepsilon,
\]

where the probability is taken over the choice of \(r\) and the random coins of \(T_C\) and \(A\), \(x \xleftarrow{\$} A(W)\) denotes the assignment to \(x\) of the output of the randomized algorithm \(A\) when run on input \(W\) with fresh random coins and \(r \xleftarrow{\$} \{0, 1\}^\rho\) denotes the random generation of an element \(r \in \{0, 1\}^\rho\) using the uniform distribution. We refer the reader to previous literature [3, 17] for more details.

3. THE DEFINITIONS AND THE MODEL

3.1 Syntax and Secrecy

In this section we will describe the definitions for our attribute-based traitor tracing scheme. A user will be identified by a pair \((\omega, id)\), where \(\omega\) is a set of attributes and \(id\) denotes the user’s identity. In our setting, it is allowable for different users with the same set of attributes. However, we require that a user should be assigned only one set of attributes. Formally, an attribute-based traitor tracing ABTT scheme consists of five polynomial-time algorithms:

**Setup:** This algorithm takes as input the security parameter \(\kappa\) and generates a set of domain parameters consisting of a master public key \(mpk\) and a master secret key \(msk\). It is a randomized algorithm.

**Key Extraction:** Given the master secret key \(msk\) and a pair \((\omega, id)\), this algorithm generates a user secret key \(d_{(\omega, id)}\). It could be probabilistic.
**Encryption:** On inputs of the master public key $mpk$, a set of attributes $\omega$ and a message $m$, this algorithm outputs a ciphertext $C$. It should be probabilistic.

**Decryption:** On inputs of a user secret key $d(\omega,id)$ and a ciphertext $C$, this algorithm outputs a plaintext message $m$, or $\bot$ to indicate a decryption error.

**Traitor Tracing:** This algorithm has oracle access to a “pirate” decryption box $D$ for a set of attributes $\omega$. It takes as input the master secret key $msk$, $\omega$, and outputs a set of user identities (called “traitors”).

Note that we define a “pirate” decryption box $D$ for a set of attributes $\omega$. This means that $D$ decrypts a non-negligible fraction of random ciphertexts encrypted under the attributes $\omega$.

We require that our attribute-based traitor tracing scheme is semantically secure under chosen plaintext attack. Here, we simply extend the Selective-ID model of security for Fuzzy Identity Based Encryption (the original ABE scheme) [1] to our setting.

Selective-Set model for our ABTT:

**Init:** The adversary declares a set of attributes $\omega$, that he wishes to be challenged upon.

**Setup:** The challenger runs the Setup algorithm of ABTT and gives the public parameters to the adversary.

**Phase 1:** The adversary is allowed to issue queries for private keys for many pairs $(\omega_j, id_j)$, where $|\omega_j \cap \omega_i| < d$ for all $j$. There is no limitation on $id_j$.

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

**Phase 2:** Phase 1 is repeated.

**Guess:** The adversary outputs a guess $b'$ of $b$.

The advantage of an adversary $A$ in this game is defined as $Pr[b' = b] - 1/2$. We say that the attribute-based traitor tracing scheme is secure in the selective model if all polynomial-time adversaries have at most a negligible advantage in the above game.

### 3.2 Traceability

We now consider the notion of traceability for our attribute-based traitor tracing scheme. We mainly refer to the definition for the public key setting in [16] and the identity-based setting in [3]. Here, we describe the definition of traceability in a slightly weaker model of security known as the selective model, in which the adversary declares which set of attributes he will create a pirate decoder before the global public parameters are generated. Our scheme is proved secure in this weak model.

Let $c, \kappa$ be two security parameters associated to the experiment.
Init: The adversary declares a set of attributes $\omega$, which he wishes to be challenged upon.

Setup: The challenger runs the Setup algorithm of ABTT and gives the public parameters to the adversary.

Key Extraction Query: The adversary is allowed to issue queries for private keys for many pairs $(\omega_j, id_j)$.

Generate Pirate Decoder: The adversary outputs a pirate decoder $D$ for $\omega$, which is the description of a probabilistic circuit that takes as input ciphertexts and outputs messages.

Tracing: The challenger runs the tracing algorithm with black-box access to $D$ to obtain a set of user identities, $S$.

We also assume that all “pirate” decryption boxes are resettable [3], meaning that they retain no state between decryptions. In particular, pirate boxes cannot self-destruct.

If we let $T$ denote the set of user identities that the adversary submitted to the key extraction query in combination with $\omega_j$, where $|\omega_j \cap \omega| \geq d$, then we say that the adversary wins the game if the following conditions hold:

- The decryption box decrypts a non-negligible fraction of random ciphertexts encrypted under the set of attributes $\omega$.
- Either $S = \emptyset$, or $S \neq T$ and $S \subset T$.
- The adversary queried the key extraction oracle at most $c$ different user identities to which the assigned set of attributes satisfy $|\omega_j \cap \omega| \geq d$. For $|\omega_j \cap \omega| < d$, we do not restrict the number of queries.

The advantage of $A$ in breaking the traceability of the ABTT scheme is defined as its probability of winning the above game. We say that ABTT is $c$-TRA-SS-CPA secure, if this advantage is a negligible function in $\kappa$ for all adversaries $A$ with running time polynomial in $\kappa$.

4. OUR SCHEME

We now describe our attribute-based traitor tracing scheme. The construction mainly borrows ideas from the second construction of the attribute-based encryption schemes proposed by Sahai-Waters [1] and the identity-based traitor tracing scheme [3].

4.1 Description

In our scheme, we also use all elements of $\mathbb{Z}_p^*$ as the universe of attributes. We assume $n_1$ be the maximum size attribute set for a user and $d$ be the threshold value. Let $U = \{id_1, id_2, \ldots, id_N\}$ be the set of all users, $s$ be the size of the symbol alphabet and codeword be a string of $\ell$ symbols. We also introduce a randomly chosen permutation on $\{1, 2, \ldots, N\}$ as in [3], denoted $\pi$. Then we associate the codeword $w^{\pi_{id}}$ with the user $id$, where $r$ is the randomness for collusion-secure codes. The trusted authority maintains the
mapping between individual users and their codewords. We use the natural encoding of symbols as bit strings of size \( \log_2 s \) for non-binary alphabets, so that codewords are represented by bit strings of size \( n_2 = \log_2 s \times \ell \). In addition, we can apply a collision-resistant hash function \( h : \{0, 1\}^r \rightarrow \mathbb{Z}_p^* \), which allows arbitrary strings as attributes.

Define the Lagrange coefficient \( \Delta_{k,S} \) for \( k \in \mathbb{Z}_p \) and a set, \( S \), of elements in \( \mathbb{Z}_p \):

\[
\Delta_{k,S}(x) = \sum_{j \in S, j \neq k} \frac{x - j}{k - j}
\]

Let \( cw \in \{0, 1\}^{s_2} \) be the bit string for a codeword and \( cw_j \) denote \( j \)th bit of \( cw \). We define a \((n_2 + 1)\)-length vector \( V = (v_i) \), \( i \in \{0, 1, \ldots, n_2\} \), whose elements are chosen at random from \( G_2 \). Let \( u_i = \phi(v_i) \) and \( \mathcal{U} \) denote the image of the vector \( V \) under the isomorphism \( \phi \), i.e., \( \mathcal{U} = (u_i) \).

We use vector \( V \) to define Waters’ hash function [2]:

\[
H(cw) = v_0 \prod_{i \in B} u_i \text{, where } B \text{ is a set of all } i \text{ for which } cw_i = 1.
\]

We also define:

\[
G(cw) = u_0 \prod_{i \in B} u_i \text{ to simplify notation. Note that } G(cw) \text{ can be computed either from vector } \mathcal{U} \text{ directly, or from vector } V \text{ using the isomorphism } \phi.
\]

Our construction follows.

**Setup:** The system parameters are generated as follows. We first generate a set of pairing groups as above at the security level \( \kappa \), along with the vectors \( V \) and \( \mathcal{U} \). We choose a random element \( h \) in \( G_2 \) and let \( g = \phi(h) \). A secret \( \alpha \in \mathbb{Z}_p \) is chosen at random, and we set \( g_1 = g^\alpha \) and \( h_1 = h^\alpha \). We require a second random element \( h_2 \in G_2 \) and let \( g_2 = \phi(h_2) \). Next, let \( N \) be the set \( \{1, 2, \ldots, n_1 + 1\} \), we choose \( t_1, \ldots, t_{n_1+1} \) uniformly at random from \( G_2 \), and compute \( c_i = \phi(t_i) \) for \( i \in N \). Now we define function \( T \) as in [1]:

\[
T(x) = h_2^{s_1} \prod_{i=1}^{n_1+1} t_i^{\Delta_{i,S}(x)},
\]

and let \( K(x) = \phi(T(x)) \), which is defined as:

\[
K(x) = g_2^{s_1} \prod_{i=1}^{n_1+1} c_i^{\Delta_{i,S}(x)} = \phi(T(x)).
\]

Finally, we choose the secret random permutation \( \pi \) and the secret randomness \( r \in \{0, 1\}^\rho \) for the code \( C \). We set the master public key and the master private key as:

\[
mpk = \{g, g_1, h_2, \mathcal{U} = (u_0, \ldots, u_{n_1}), c_1, \ldots, c_{n_1+1}\}, \text{ msk} = \{\alpha, h, V, t_i, \ldots, t_{n_1+1}\}.
\]

**Key Extraction:** This algorithm allows a user with a set of attributes \( \omega \) and an identity \( id \) to obtain a decryption key. With this decryption key, the user can decrypt a message encrypted under a set of attributes \( \omega' \), if and only if \( |\omega' \cap \omega| \geq d \). Let \( cw \) be the codeword corresponding to identity \( id \). The key distribution center first chooses a \( d - 1 \) degree polynomial \( q \) at random such that \( q(0) = \alpha \), and two sets \( \{r_i\}_{i \in \omega} \) and \( \{r'_i\}_{i \in \omega} \) where the elements \( r_i \) and \( r'_i \) are randomly selected from \( \mathbb{Z}_p \) for all \( i \in \omega \). The private key that needs to be sent to the user will consist of three sets. The first set is \( \{D_i\}_{i \in \omega} \), where the elements are constructed as:
\[ D_i = h^{d(i)} T(i) H(cw')^i. \] (3)

The second set is \( \{d_i\}_{i \in \omega} \) where the elements are constructed as \( d_i = h^i \). The last set is \( \{d_i'\}_{i \in \omega} \) with \( d_i' = h^{i'} \).

**Encryption:** A message \( M \) is defined as an element in \( G_T \). To encrypt \( M \) using attribute set \( \omega' \), the sender first chooses a random value \( t \in Z_p \). The ciphertext is then computed as:

\[
C = (\omega', C_1 = g^t, C_2 = M{h}e(g, h_1)^t, C_3 = \{B_i = K(i)\}_{i \in \omega'}, C_4 = (u_i^j)_{j = 0, \ldots, s}). \quad (4)
\]

**Decryption:** Let \( C \) be a valid encryption of \( M \) under \( \omega' \). \( C \) can be decrypted by a user with the private key for \( \omega \), where \( |\omega' \cap \omega| \geq d \). First, compute:

\[
\prod_{j \in B} e(B_j, d_i) e(G(cw')^i, d_i')^\Delta = 1. \quad (5)
\]

where \( B \) is a set of all \( j \) for which \( cw_j = 1 \). Then choose an arbitrary \( d \)-element subset, \( S \), of \( \omega' \cap \omega \). The ciphertext can be decrypted as:

\[
M = C_2 \prod_{i \in \omega} \left( e(B_i, d_i) e(G(cw')^i, d_i')^{\Delta_i} \right)^{\Delta_i x(0)}. \quad (6)
\]

**Traitor Tracing:** The tracing step can only be done by the key distribution center as in [3]. This algorithm has access to a pirate box \( D \) for \( \omega \). \( D \) correctly decrypts ciphertexts encrypted under \( \omega \) with probability \( \delta(\kappa) \), which is a non-negligible function of \( \kappa \). For convenience, we let \( C^{(i)}_4 \) denote the \( (\lfloor \log_2 s(k - 1) \rfloor + j) \)th element of \( C_4 \). For each \( 1 \leq i \leq \ell \) and \( 1 \leq j \leq \lfloor \log_2 s \rfloor \), initialize counter \( c_{tr_i,j} \) = 0 and run the following test \( n = 16\kappa/\delta(\kappa) \) times:

1. Choose a random message \( m \).
2. Encrypt \( m \) under \( \omega \) to form a ciphertext \( C = (\omega, C_1, C_2, C_3, C_4) \).
3. Replace \( C^{(i)}_4 \) with a random element from \( G_1 \).
4. Query the pirate decoder \( D \) on the altered ciphertext \( C \).
5. If the decoder outputs the message \( m \) then increase \( c_{tr_i,j} \).

After these iterations, reconstruct the bit string \( cw' \) of length \( n_2 \) as follows. Let \( cw'_{i,j} \) denote the bit of \( cw' \) at position \( \lfloor \log_2 s(k) \rfloor + j \). Set \( cw'_{i,j} = 1 \) if \( c_{tr_i,j} \leq 4\kappa \) or set \( cw'_{i,j} = 0 \) otherwise. Next, decode the bit string \( cw' \) as a symbol string \( x \) of length \( \ell \), choosing any symbol if the corresponding bit string is not a valid encoding of a symbol in \( \Sigma \). Finally, use the tracing algorithm of the code to compute \( I \leftarrow \pi \text{ Tr}(x, r) \). Let \( r \) be the reverse permutation of \( \pi \), the algorithm returns the identity set of traitors \( id_{\omega(k)} \) for \( k \in I \).

### 4.2 Other Constructions

The above construction is based on the large universe construction of Sahai-Waters’ attribute-based encryption scheme. However, we notice that all methods used in previous construction are general in some situations, *i.e.*, by using these technologies we can con-
ATTRIBUTE-BASED TRAITOR TRACING

Here we give a simple description of the construction based on the first construction from [1]. Let $P$ be the universe attributes set.

**Setup:** We first generate a set of pairing groups as above at the security level $\kappa$, along with the vectors $V$ and $U$. We choose a random element $h \in \mathbb{G}_2$ and let $g = \phi(h)$. A secret $\alpha \in \mathbb{Z}_p$ is chosen at random, and we set $g_1 = g^\alpha$ and $h_1 = h^\alpha$. We require a second random element $h_2 \in \mathbb{G}_2$ and let $g_2 = \phi(h_2)$. Next, choose $t_1, \ldots, t_{|P|}$ uniformly at random from $\mathbb{Z}_p$.

The published public parameters are: $T_1 = g_1 t_1, \ldots, T_{|P|} = g_1 t_{|P|}$, $R = \hat{e}(g, h_2)^\alpha$ and $g$, $U$. The master key is: $t_1, \ldots, t_{|P|}, h, h_2, \alpha, V$. Finally, we choose the secret random permutation $\pi$ and the secret randomness $r \in \{0, 1\}^\rho$ for the code $C$.

**Key Extraction:** Let $cw$ be the codeword corresponding to identity $id$. The key distribution center first chooses a $d-1$ degree polynomial $q$ at random such that $q(0) = \alpha$, and a set $\{r_i\}_i \in \omega$ where the element $r_i$ randomly selected from $\mathbb{Z}_p$ for all $i \in \omega$. The private key is: $D_i = \frac{h_{\pi(i)} H(cw)}{t_{\pi(i)}} r_i, d_i = (h_i)^r$ for all $i \in \omega$.

**Encryption:** The sender chooses a random value $t \in \mathbb{Z}_p$. For $\omega'$ the ciphertext is computed as:

$$C = (\omega', C_1 = MR^t, C_2 = \{T_i\}_i \in \omega', C_3 = (u_j)_{j=0,\ldots,n_2}).$$

**Decryption:** First compute $G(cw)$. Then choose an arbitrary $d$-element subset, $S$, of $\omega' \cap \omega$. The ciphertext can be decrypted as:

$$M = C_1 \prod_{i \in S} \left( \frac{\hat{e}(G(cw), d_i)}{\hat{e}(T_i, D_i)} \right)^{\lambda_{\pi(i)} (0)}.$$

**Traitor Tracing:** The same as in section 4.1.

5. SECURITY PROOFS

We now present the proofs of security to show that the resulting system is a secure attribute-based traitor tracing system.

**Theorem 1** The advantage of an adversary in the selective-set game is negligible in our attribute-based traitor tracing scheme under the decisional BDH assumption.

**Proof:** The proof of this theorem falls along the lines of the proof of Theorem 2 in [1]. There are some differences between [1] and our setting. However, one notices that our construction makes use of the asymmetric version of Sahai-Waters’ scheme. Fortunately, this difference does not seem to create a problem for the construction of simulator.

Suppose there exists a polynomial-time adversary, $A$, that can attack our scheme in
the Selective-Set model with advantage $\varepsilon$. We build a simulator $\mathcal{B}$ that can play the Decisional BDH game with advantage $\varepsilon/2$. In our setting, the simulator will take BDH challenge $(h, A = h^a, B = h^b, C = h^c, Z)$, where $Z \in G_T$, and outputs a guess, $\mu$, as to whether the challenge is a BDH tuple.

We first let the challenger set the groups $G_1$, $G_2$ and $G_T$ with an efficient asymmetric bilinear pairing $\hat{e}$. Let $g$ and $h$ be generators of $G_1$ and $G_2$, respectively, and let $\varphi : G_2 \to G_1$ be an efficiently computable isomorphism such that $\varphi(h) = g$. The challenger flips a fair binary coin $\mu$ outside of $\mathcal{B}$’s view. If $\mu = 0$, the challenger sets $(A, B, C, Z) = (h^a, h^b, h^c, \hat{e}(g, h)^z)$; otherwise it sets $(A, B, C, Z) = (h^a, h^b, h^c, \hat{e}(g, h)^{\mu z})$ for random $a, b, c, z$. The simulator sets a code $C$ and proceeds as follows:

- $\mathcal{B}$ will run $\mathcal{A}$ and receive the challenge $\omega i$, which is an $n_1$ element set of members of $Z_p$. The simulator assigns $h_1 = A$ and $h_2 = B$. It then chooses an $(n_2 + 1)$-length vector, $Y = (Y_1, \ldots, Y_{n_2})$, where the elements of $Y$ are chosen at random in $Z_p$ for $i \in \{0, 1, \ldots, n_2\}$ and sets $\mathcal{V}$ as $v_i = h^{Y_i}$. Additionally, it chooses a random $n_1$ degree polynomial $f(x)$ and calculates an $n_1$ degree polynomial $u(x)$ such that $u(x) = -x^{v_i}$ for all $x \in \mathcal{V}$ and where $u(x) \neq -x^{v_i}$ for some other $x$. Now the simulator assigns the public parameters $U$ as $u_i = \varphi(v_i)$, $c_j = \varphi(h_j^x h_1^{x^0})$ for $j \in \{1, 2, \ldots, n_1 + 1\}$ and $g_1 = \varphi(h_1)$, and then gives them to $\mathcal{A}$ with the parameters $g$, $h_2$. Note that we implicitly have $T(x) = h_2^{c_1 u(x) + c_2 x^0}$ and $K(x) = \varphi(T(x))$. In addition $\mathcal{B}$ can compute $H(cw)$ and $G(cw)$ as the definitions.

- Suppose $\mathcal{A}$ requests a private key $(\omega_i, id)$. Note that $\mathcal{A}$ can query on arbitrary pairs $(\omega, id)$, where $|\omega \cap \omega| < d$. Let $cw$ be the codeword corresponding to identity $id$ (note that $cw$ can also be the binary representation of identity $id$ in this proof) and the length of $cw$ is $n_2$. First define three sets $\Gamma = \omega \cap \omega$, $\Gamma'$ such that $\Gamma \subseteq \Gamma' \subseteq \omega$ and $|\Gamma'| = d - 1$, and $S = \Gamma' \cup \{0\}$. Next, we define $D_0, d_i$ and $d'_i$ for $i \in \Gamma'$ as: $D_0 = h_1^{\omega} T(i) H(cw)^{v_i}$, $d_i = h^{v_i}$, $d'_i = \hat{e}(h_i, d_i)$, where $\lambda_*, r_i$ and $r'_i$ are chosen randomly in $Z_p$. We have chosen a random $d - 1$ degree polynomial $q(x)$ by choosing its value for the $d - 1$ points randomly by setting $q(i) = \lambda_i$ in addition to having $q(0) = a$. The simulator calculates the decryption key values for all $i \in \omega - \Gamma'$ as:

$$
D_i = (\prod_{j \in \Gamma'} h_2^{x^{d_i}(j) f(j) + c_i u(j) h^{c_i x^{d_i}(j)} f(j)} H(cw)^{v_i})^{h_i^{x^{d_i}(j)}},
$$

$$
d_i = (h_2^{c_i u(j) h^{v_i}})^{d_i^{x^{d_i}(j)}} \text{ and } d'_i = h^{c_i x^{d_i}(j)} h^{x^{d_i}(j)}, \text{ where } r_i \text{ and } r'_i \text{ are chosen randomly in } Z_p.\]$$

- The simulator flips a fair binary coin, $\nu$, and returns an encryption of $M_e$. The challenge ciphertext is:

$$
C' = (\omega i, \varphi(C), C_2^* = M_e, Z, \{B_{ji} = (\varphi(C))^{y_j}(\varphi(C))^{y_j}(\varphi(C))^{y_j} \}_{j = 0, \ldots, n_2}).
$$

If $\mu = 0$, then the ciphertext is a valid ciphertext for the message $M_e$ under $\omega i$. Otherwise, $C_2^*$ will be a random element of $G_2$ from the adversaries view. This element gives no information about $M_e$.

- The simulator acts exactly as it did in Phase 1.
Lemma 1  If \( \text{cw}_{i,j} = 0 \) in the codewords of all users in the collusion \( C \), then \( D \) correctly decrypts a random ciphertext that has been tampered with at position \((i', j')\) with probability \( p_0 \geq \delta(\kappa) - \text{Adv}^{\text{ddhp}}_{\text{R,\varepsilon}}(\kappa) \).

Proof: Let \( A \) be an adversary against the tracing property of our scheme. Let \( D \) be the decryption box of \( A \) produced that correctly decrypts random ciphertexts that have been tampered with at position \((i', j')\) with probability \( p_0 \leq \delta(\kappa) - \gamma \) for some \( \gamma > 0 \). We will construct an algorithm \( B_1 \) which uses \( A \) to solve the DDH problem in \( G_1 \) with an advantage \( \gamma \).

The algorithm \( B_1 \) will take DDH challenge \((A = g^a, B = g^b, Z)\). Let \( k = s(i' - 1) + j' - 1 \). \( B_1 \) run \( A \) and receive the challenge \( \omega' \). It then chooses two exponents \( \alpha, \beta \in \mathbb{Z}_p \) and sets \( g_1 = g^\alpha, h_1 = h^\alpha, h_2 = h^\beta \). It chooses an \((n_2 + 1)\)-length vector, \( Y = (v_1) \), where the elements of \( Y \) are also chosen at random in \( \mathbb{Z}_p \) for \( i \in \{0, 1, ..., n_2\} \). It sets \( v_k = 1, u_k = A \) and \( v_i = h^\beta, u_i = g^\beta \) for \( i \neq k \). Additionally, let \( N \) be the set \( \{1, 2, ..., n_1 + 1\} \). It chooses an \((n_1 + 1)\)-length vector \( X = (x_i) \), where the elements of \( X \) are also chosen at random in \( \mathbb{Z}_p \) and sets \( c_j = g^\beta \) for \( j \in N \). \( B_1 \) gives the public parameters \( mpk: g, g_1, h_2, U, c_j \) (for \( j \in N \)) to \( A \).

It also chooses secret randomness \( r \in \{0, 1\}^d \) for the collusion-secure code.

Assuming \( A \) query a private key for \((\omega, \text{id})\) (\( A \) can query at most \( c \) different user identities to which the assigned set of attributes satisfy \( |\omega' \cap \omega| \geq \delta \). For \( |\omega' \cap \omega| < \delta \), we do not restrict the number of queries). Let \( \text{cw} \) be the codeword corresponding to identity \( \text{id} \). \( B_1 \) responds to the key extraction queries as follows (\( B_1 \) knows \( \text{cw}_\omega = 0 \)). It chooses a random \( d - 1 \) degree polynomial \( q(x) \) with \( q(0) = \alpha \), then sets

\[
D_i = h^\beta (i) T(i) Y H(\text{cw})^r_i, \quad d_i = h^\beta, \quad d'_i = h^\beta
\]

where \( r_i \) and \( r'_i \) are chosen randomly in \( Z_p \) for all \( i \in \omega \). Note that \( B_1 \) can compute \( H(\text{cw}) \), even though it does not know \( \text{vk} \).

At the end of this stage \( A \) will output a pirate decoder \( D \) for \( \omega' \). Algorithm \( B_1 \) then generates a random message \( m \) and computes the ciphertext

\[
(\omega'; B; \hat{m}(B, h_2)^\alpha; \{B^\beta j^2 \prod_{j=1}^{n_1+1} B^3 A_h n^{(j)} \}_i \mid j \in \omega'; \{B^{c_j} \}_i \mid \omega \cap \omega, i \in k, \{Z \}_i \in \mathbb{Z})
\]
It pass the ciphertext to the decoder $\mathcal{D}$. Algorithm $B_1$ outputs 1 if the decoder correctly decrypts $m$, or outputs 0 otherwise.

If $Z = g^s$, then $B_1$ outputs 1 with probability $\delta(\lambda)$. If $Z$ is random, then $B_1$ outputs 1 with probability at most $\delta(\lambda) - \gamma$. The advantage of $B_1$ in solving DDH problem is that $\text{Adv}_{B_1, G_1}^\text{ddh}(\lambda) \geq \gamma$, from which the lemma follows.

**Lemma 2** If $\text{cw}_{i,j} = 1$ in the codewords of all users in the collusion $C$, then $\mathcal{D}$ correctly decrypts a random ciphertext that has been tampered with at position $(i', j')$ with probability $p_i \geq \text{Adv}_{B_2, G_2}^\text{cbdhp}(\lambda)$.

**Proof:** Let $A$ be an adversary against the tracing property of our scheme. Let $\mathcal{D}$ be the decryption box of $A$ produced that correctly decrypts random ciphertexts with probability $\delta(\lambda)$, but that correctly decrypts ciphertexts that have been tampered with at position $(i', j')$ with probability $p_i$. We will construct an algorithm $B_2$ which uses $A$ to solve the CBDH problem.

The algorithm $B_2$ will take CBDH challenge $(A = h^a, B = h^b, C = h^c)$. Let $k = s(i' - 1) + j' - 1$. $B_2$ will run $A$ and receive the challenge $\alpha$. It first sets $g_1 = \phi(A)$ and $h_2 = B$. It chooses an $(m + 1)$-length vector, $Y = (y_i)$, where the elements of $Y$ are also chosen at random in $\mathbb{Z}_p$ for $i \in \{0, 1, \ldots, n_2\}$. It sets $\nu = h_1^{2\gamma a}$, $u_k = \phi(v_k)$ and $v_j = h^b$, $u_i = g^{\gamma}$ for $i \neq k$. Additionally, it chooses a random $n_1$ degree polynomial $f(x)$ and calculates $n_1$ degree polynomial $u(x)$ such that $u(x) = -x^j$ for all $x \in \alpha$ and where $u(x) \neq -x^j$ for some other $x$. Now the simulator assigns $c_j = \phi(h_1^{j}h^i)$ for $j \in \{1, 2, \ldots, n_1 + 1\}$, and then gives the public parameters to $A$.

Assuming $A$ queries a private key for $(\omega, id)$. Let $cw$ be the codeword corresponding to identity $id$. In order to respond the query, the following sets are considered. Firstly, define set $\Gamma = \omega \cap \alpha$. Secondly, define set $\Gamma'$ such that $\Gamma \subseteq \Gamma' \subseteq \omega$ and $|\Gamma'| = d - 1$ if $|\omega \cap \alpha| < d$, or define set $\Gamma'$ such that $\Gamma' \subseteq \Gamma$ and $|\Gamma'| = d - 1$ otherwise. Finally, define set $S = \Gamma' \cup \{0\}$. Now $B_2$ computes $D_i, d_i, d_i'$ for $i \in \Gamma'$ as:

$$D_i = h^\lambda T(i)\nu H(cw)^{\nu'}, d_i = h^{\nu'}, d_i' = h^{\nu'},$$

where $\lambda$, $r$, and $r'$ are chosen randomly in $\mathbb{Z}_p$. We have chosen a random $d - 1$ degree polynomial $q(x)$ by choosing its value for the $d - 1$ points randomly by setting $q(i) = \lambda_i$ in addition to having $q(0) = a$. The simulator calculates the decryption key values for all $i \in \omega - \Gamma'$ as:

$$D_i = \left( \prod_{i \in \omega} h^\lambda \frac{T(i)^{\nu} \cdots (i)}{\nu} (h^\lambda)^{\nu} (h^\lambda)^{\nu'} (h^{\nu} h^{\nu'})^{\nu s} \prod_{t \in \omega, t \neq k} (h^{\nu})^{\nu} \right)^{\nu s \cdot (i)},$$

$$d_i = h^\lambda \frac{T(i)}{\nu} \text{ and } d_i' = h^{(r - r') \nu},$$

where $r$ and $r'$ are chosen randomly in $\mathbb{Z}_p$.

At the end of this stage $A$ will output a pirate decoder $\mathcal{D}$ for $\alpha$. Algorithm $B_2$ chooses randomly $Z \in G_1$, $C_2 \in G_T$ and generates the challenge ciphertext with:

$$\omega'; \phi(C); C_2; \{(\phi(C)^{\nu})_{i \in \alpha}'; \phi(C)^{\nu s} \}_{t \in \omega, t \neq k} (Z)_{t \neq k}.$$}

The pirate decoder $\mathcal{D}$ with this ciphertext will output the corresponding plaintext $m$.
with probability $p_1$, as if $Z$ were chosen correctly as $u_k^c$. In this case $B_3$ can compute $\hat{e}(g, h)^{\alpha_k}$ by $C_2/m$. So $B_3$ has an advantage $\text{Adv}^{\text{chdh}}_{B_2, G_2}(\kappa) \geq p_1$.

**Theorem 2** Let the underlying code be $(c, N, \varepsilon)$ collusion-secure code and its length is $\ell$ over an alphabet of size $s$. Assuming that the DDH problem in $G_1$ is hard and the CBDH problem in $G_2$ is hard. Our ABTT scheme described above is $c$-TRA-SS-CPA secure. More specifically, let $A$ be any polynomial-time adversary. If $A$ uses the keys of a collusion of at most $c$ users and builds an untraceable decoder that correctly decrypts a fraction $\delta(\kappa)$ of ciphertexts, then the advantage of $A$ is at most

$$\text{Adv}^{\text{TRA-SS-CPA})}_{A, ABTT}(\kappa) \leq \epsilon + \ell \left\lceil \log_2 s \right\rceil (\text{Adv}^{\text{chdh}}_{B_2, G_2}(\kappa) + e^{-\kappa}),$$

wherever $\delta(\kappa) \geq 2 \text{Adv}^{\text{chdh}}_{B_2, G_2}(\kappa)$ where $B_1, B_2$ are polynomial-time algorithms depending on $A$ and $e$ is the base of the natural logarithm.

**Proof:** It is easy to see that this theorem follows from the above two lemmas and the proof of Theorem 1 in [3] with small modification for our setting. Here, we give a sketch of the proof. Let $A$ be an adversary against the tracing property of our scheme. We use $A$ to construct an algorithm $A'$ against the tracing property of the collusion secure code. The proof shows that if $A$ successfully avoids being traced with high probability in our scheme, then $A'$ will output a codeword $x$ that cannot be traced.

Let $A'$ mount an attack for the collusion $C = \{i_1, \ldots, i_c\}$, where $i_1, \ldots, i_c$ are randomly selected from $\{1, \ldots, N\}$. Let the corresponding codewords be $W = w_j$: $j = 1, \ldots, c$. $A'$ runs $A$ and receives $\omega$. Assuming $A$ query a private key for $(\omega, id)$. If $|\omega \cap \omega| \geq d$ (at most $c$ different user identities), $A'$ assigns $w_j$ to the $j$th such query and responds as normal. If $|\omega \cap \omega| < d$, $A'$ assigns a random $cw$ from $C/W$ to the $id$ and responds as normal. After that, $A$ outputs $\square$ and terminates. $A'$ then runs the traitor tracing algorithm of our scheme and outputs the value of the symbol string $x$.

In [3], the authors use a Chernoff bound to analyze the overall success probability of the tracing algorithm. Their results are also applicable for our setting. The $16\kappa/\delta(\kappa)$ iterations are required. Their proof shows that the output $x$ does not in the feasible set $FS(W)$ with probability at most

$$\Pr[x \not\in FS(W)] \leq \left\lceil \ell \log_2 s \right\rceil (\text{Adv}^{\text{chdh}}_{B_2, G_2}(\kappa) + e^{-\kappa}).$$

Now the theorem follows from the above result.

**6. CONCLUSION**

We introduce the notion of attribute-based traitor tracing and describe a construction based on collusion secure codes. This notion is a feasible method to mitigate the key misuse problem in the scenarios of attribute-based encryption. Unfortunately, our scheme is not practical due to the sizes of public key and ciphertext. In addition we only prove the traceability property of our traitor tracing system in a slightly weaker model known as the selective model. Thus it is interesting to construct an ABTT system that satisfies the traceability property in the full model.
REFERENCES


**Yong-Tao Wang (王永涛)** received his M.S. degree in Computer Science and Engineering from Xihua University, Chengdu, 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 Jiaotong 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.

**Jian-Hong Chen (陈剑洪)** 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 Jiaotong University. His research interests include public key cryptosystem and pairing based cryptosystem.