Making *Any* Attribute-Based Encryption Accountable, Efficiently

Junzuo Lai[1,2](✉) and Qiang Tang[3]

[1] Ji'nan University, Guangzhou, China
laijunzuo@gmail.com
[2] State Key Laboratory of Cryptology, Beijing, China
[3] New Jersey Institute of Technology, Newark, USA
qiang@njit.edu

Abstract. Attribute-based encryption (ABE) as one of the most interesting multi-recipient public encryption systems, naturally requires some "tracing mechanisms" to identify misbehaving users to foster accountability when unauthorized key re-distributions are taken place.

We give a generic construction of (black-box) traceable ABE which only doubles the ciphertext size of the underlying ABE scheme. When instantiating properly, it yields the first such scheme with constant size ciphertext and expressive access control.

Furthermore, we extend our generic construction of traceable ABE to support authority accountability. This property is essential for generating an un-deniable proof for user misbehaviors. Our new generic construction gives the first black-box traceable ABE with authority accountability, and constant size ciphertext. All properties are achieved in standard security models.

1 Introduction

Attribute-Based Encryption (ABE), first introduced in [12,29], naturally generalizes the concept of identity based encryption (IBE) to support more expressive "identities" as they can be any string. Two major types of attribute based encryption schemes exist: ciphertext-policy attribute based encryption (CP-ABE) [2] and key-policy attribute based encryption (KP-ABE) [12]. In a CP-ABE scheme, each ciphertext is associated with a decryption policy which can be represented using e.g., an access structure or a boolean formula; every user's decryption key is associated with an attribute set which is used to describe the key owner. A user is able to decrypt a ciphertext only if the set of attributes associated with the user's decryption key satisfies the decryption policy associated with the ciphertext. While in a KP-ABE scheme, the situation is reversed, where every ciphertext is associated with a set of attributes and every user's decryption key is associated with an access structure.

Since its introduction, great advancements have been taken place over the years, both on the expressibility of the decryption policy (can be as general as an circuit [8,9]), and on the asymptotic efficiency (e.g., constant size ciphertext).

J. Lopez et al. (Eds.): ESORICS 2018, LNCS 11099, pp. 527–547, 2018.
https://doi.org/10.1007/978-3-319-98989-1_26

Due to its expressibility, attribute based encryption could be very useful in many settings, here we list two typical application scenarios: (i) enforcing access control by encrypting the data with the access control policy, and issuing decryption keys to users according to their attributes. Such mechanism can be used in company internal networks to improve the robustness of their access control functionality. (ii) distributing contents via a cloud or content delivery network. The content provider simply encrypts the data and store the ciphertext in the publicly accessible cloud, then he issues decryption keys for each subscriber according to his subscription package. For instance, a movie producer encrypts two versions of a movie (based on the resolutions, say) m_1, m_2. The decryption policy for the ciphertext corresponding to the standard quality version m_1 (say 720p) is "(status = regular user $\vee$ status = premier user) $\wedge$ age $\geq$ 18" and the decryption policy for the high resolution version m_2 (say 1080p) is "status = premier member $\wedge$ age $\geq$ 18". During the subscription, a premier subscriber who paid higher fees can obtain decryption key that allows him to have access to the high resolution version of the movie.

Despite all recent advancements and those potential applications, attribute based encryption schemes still have not been widely deployed in practice. Besides the potential problem of its concrete efficiency, there is another serious accountability problem that needs to be addressed before the deployment of attribute based encryption (at least to the above two application scenarios). We can see that attribute based encryption is a special kind of multi-recipient encryption scheme in which the decryption capability (or the attributes) from different users may overlap. Imagine in the access control example, each employee in a company is assigned with a secret key according to his position (and the corresponding access rights) to get access to the company documents which may contain business secret. A natural key management policy of the company could be "do not share your decryption key to others, especially to outsiders". But there is no way to prevent a "corrupted" employee from doing so. Although an employee may not directly expose her decryption key material, she can still write a decoder program and share a potentially more restricted decoder program to others. When such an unauthorized decoder, which can be used to decrypt ciphertext with certain policies, is noticed, there are multiple key owners might be suspects. Thus identifying the source of such unauthorized sharing is critical for the manager to carry out proper punishments in order to enforce his key management policy. Similar issue arises in the second application scenario that a pirate decoder of movies should be trace back to the misbehaving subscriber.

The first accountability property we will pursue is to enable tracing[1] from an unauthorized decoder to the actual key owner. From the first look, such traceability in ABE schemes seems to be very close to that in traitor tracing schemes [6]. The major difference here is that in a traitor tracing scheme, every user has the same privilege, just finding out one corrupted user is a reasonable goal.

[1] We note here that the tracing mechanism we are pursuing in this paper follows a similar vein to the well-known notion of traitor tracing [6], and it is not hard to see that completely preventing the unauthorized leakage is essentially infeasible.

While in the setting of ABE, different user may have different access right, just identifying one corrupted user is not satisfying. For example, in the movie distribution application, the two users, one with attribute "status = premier member" and the other with attribute "status = regular member", collude to produce a pirate decoder that has attribute "status = premier member", but the tracing algorithm may only return the user with attribute "status = regular member", who has less to lose. (Such difference was first pointed out and formalized by Katz and Schroder in [13] in the setting of predicate encryption.) It turns out that trivially combining ABE with a traitor tracing scheme would only achieve above weak traceability and fail for the stronger traceability requirement. (see details at the end of Sect. 3.1). A series of works tried to combine traitor tracing and ABE in a complex way for achieving stronger traceability.

As a result, they will have to use the inefficient ABE schemes or traitor tracing constructions, thus incur large ciphertext overhead (e.g., square root of the number of users [22]); or only support very primitive policy [20], and white-box traceability [23,24]. Instead, in this paper, we demonstrate how to compile *any* ABE scheme to satisfy the stronger traceability. Such flexibility enables us to choose the best possible ABE scheme to remove both hurdles above. In particular, the size of ciphertext can be pushed down to constant.

However, having strong traceability is still not enough for resolving the accountability problem in attribute based encryption. To see this, let us continue with the above example. Suppose one employee is traced from an unauthorized decoder leaked to the competing company by the manager. When the manager shows the result and asks the employ to resign, the employee can confidently deny and claim that the tracing result is not an non-repudiable proof thus cannot be considered as an evidence for her misbehavior (even brought to a court). This is true because attribute based encryption has the key escrow problem that a key generation center is needed to issue the decryption keys. In this case, either a corrupted user who obtains a secret key from the key generation center, or a corrupted key generation center could be responsible.

The above further motivates us to consider the "accountability" on the key generation center so that an un-deniable proof can be established once a misbehaving user is caught. This is a natural generalization of accountable authority identity based encryption proposed in Crypto' 07 [10] in which every identity will correspond to exponentially many keys, and the user picks one of them obliviously. There are also works considering such a notion in ABE [25]. Unfortunately, besides relying on specific ABE construction and inherit all weakness, all those results can only work if the corrupted party (either a key owner or the key generation center) leaks a *well-formed* secret key. This obviously cannot be true in practice as the corrupted party can simply modify the key material and give instructions about how to adapt the decryption algorithm, or even write an obfuscated program so that the actual secret key used is never exposed. What's worse, this condition also puts further restriction on the definition of non-framing property, i.e., a malicious key generation center cannot frame an innocent user. In such a definition, the adversary is only allowed to run key generation protocol for one specific victim. However, in practice, a key generation center can always wait until many keys are issued, and frame one of them.

We also make progress along this line. We further compile our generic traceable CP-ABE scheme to support authority accountability which allows *black-box* tracing, i.e., the tracing algorithm only needs oracle access to an unauthorized decoder, and no artificial restriction is put on the non-framing definition.

1.1 Our Contributions

In this paper, we give a thorough study of accountability problems in attribute based encryption schemes. We give a *generic* construction of traceable ABE schemes and further make them accountable authority. Moreover, the generic construction only doubles the ciphertext size and supports black-box traceability which is *the* standard model for tracing. The benefits of such a generic construction are twofolds:

Practical benefits—if we instantiate the generic construction with an efficient ABE scheme, it gives the *first* constant size ciphertext traceable ABE scheme, also the *first* accountable authority ABE scheme with black-box traceability (still with constant size ciphertext).

Conceptual benefits—there have been various works considering ad hoc methods combining ABE schemes with traitor tracing schemes, which "obfuscate" the essence of traceability in ABE schemes. Our generic construction peels off the complexity of both its construction and analysis, and demonstrate a simple and clear picture about how accountability problems in ABE could be addressed. We use CP-ABE as an example to demonstrate our generic constructions, we note that our technique actually can easily be adapted to KP-ABE schemes. More concretely, our contributions are as follows:

1. We first propose a generic construction of traceable CP-ABE that can compile any CP-ABE to have traceability. The traceability is done in a standard black-box way that the tracing algorithm only need oracle access to the unauthorized (or pirate) decoder. Our construction utilizes a combinatorial object of fingerprinting codes, and expands the attribute set of each user with extra indices represented by the codeword that is assigned to him. If we pick the famous Tardos codes [30], our generic construction only double the ciphertext size of the underlying ABE. An overview comparing the efficiency of our traceable CP-ABE scheme to those of other traceable CP-ABE schemes is given in Table 1.
   We emphasize that such generic construction achieves the strong traceability that the accused traitor not only participates in producing the pirate decoder, but also his attributes are indeed used by the pirate decoder. See Sect. 3 for formal definitions.
2. We then further transform our generic traceable CP-ABE to be authority accountable, which means the key generation center cannot be aware of the user secret key completely, thus an un-deniable proof can be formed if a traitor is caught from a pirate decoder (this is done via a new Judge protocol). The

simple structure of our generic construction of traceable CP-ABE provides us opportunities to upgrade the construction. Inspired by the concept of asymmetric fingerprinting codes, we adapt asymmetric Tardos codes [14] to the setting of accountable authority CP-ABE.
This new generic construction for the first time allows black-box traceability while preserving the efficiency of our traceable CP-ABE. An overview comparing the efficiency of our accountable authority CP-ABE scheme to those of other accountable authority CP-ABE schemes is given in Table 2.
The main challenge in this setting is to ensure no inconsistency between the tracing and the Judge protocol, i.e., an identified traitor will evade the confirmation from the judge. This heavily relies on the security of the asymmetric fingerprinting scheme. We further utilize a technical building block called fingerprinted data transfer for the key generation protocol to ensure that no innocent user can be framed.

Table 1. Comparison of traceable CP-ABE schemes, where "N" denotes the total number of users in the system, and we instantiate our generic construction with the CP-ABE scheme proposed in [27].

Scheme	Large universe	Expressiveness of access structures	Black-box traceability	Ciphertext size
[20]	$\times$	$\times$	$\surd$	$O(1)$
[23]	$\times$	$\surd$	$\times$	$O(1)$
[24]	$\surd$	$\surd$	$\times$	$O(1)$
[22]	$\times$	$\surd$	$\surd$	$O(\sqrt{N})$
Ours	$\surd$	$\surd$	$\surd$	$O(1)$

Table 2. Comparison of accountable authority CP-ABE schemes, where we instantiate our generic construction with the CP-ABE scheme proposed in [27].

Scheme	Large universe	Expressiveness of access structures	Black-box traceability	Ciphertext size
[25]	$\times$	$\surd$	$\times$	$O(1)$
[32]	$\surd$	$\surd$	$\times$	$O(1)$
Ours	$\surd$	$\surd$	$\surd$	$O(1)$

1.2 Related Work

Traceable Attribute Based Encryption. Traceable ABE has been studied in various works [20,22–24]. To the best of our knowledge, *all* of them consider ad hoc combination of traitor tracing and specific attribute based encryption. In particular, the scheme in [20] only supports access structures having a single AND gate with wildcard. The schemes in [23,24] support only *white-box* traceability, i.e., it only works against malicious users from leaking well-formed decryption keys directly. Later in [22], Liu et al. proposed an expressive black-box traceable CP-ABE, but it incurs large ciphertext size (square root to the total number of users), thus seriously hinders its practicality. Our generic construction does not suffer from any of the above restrictions.

Accountable Authority Attribute Based Encryption. In order to mitigate the key escrow problem and the malicious key delegation problem in CP-ABE, the notion of accountable authority CP-ABE was studied in [25,32]. Both constructions only achieve *white-box* traceability, i.e., requires the adversary to provide a well-formed secret key. [25] further restricts the malicious key generation center to execute key generation protocol with only one target user. This is not only unrealistic, but also excludes the challenge in tracing systems: the collusion problem. Those serious restrictions suggest that the notion of accountable authority ABE has not been understood. Our generic construction makes a step forward.

Accountable Authority Identity Based Encryption. Goyal [10] introduced the notion of A-IBE as an approach to mitigate the key escrow problem in IBE, Subsequently, Goyal et al. [11] proposed a construction having traceability in the full black-box model with large ciphertext size. Libert and Vergnaud [21] proposed an efficient A-IBE scheme, but only is proven traceable in the weak black-box model. Sahai and Seyalioglu [28] presented the first A-IBE scheme which achieves full black-box traceability and adaptive security against dishonest users at the cost of having a linear sized ciphertext. Under non-standard assumptions, Yuen et al. [31] gave an A-IBE scheme with constant-size ciphertext, while has full black-box traceability and adaptive security against dishonest users. Lai et al. [19] proposed the first A-IBE scheme with public traceability, where tracing a decryption box only uses a public tracing key. Recently, Kiayias and Tang [18] gave a generic A-IBE scheme using oblivious transfer, and showed how to modify the generic construction to provide public traceability. However, their technique cannot be trivially extended to ABE setting due to collusion.

Self-enforcement and Proactive Deterring Mechanisms. Self-enforcement was initially proposed in digital signets that leaking a decryption key leads to revealing of some user secret [7]. Later it was systematically studied in enforcing key management policy in public key infrastructure [16], and in deterring copyright infringement [17]. Especially, leveraging the properties of cryptocurrency, [16,17] studied how to realize the deterrence that unauthorized re-distribution of pirate decoder leads to the loss of coins directly. Considering proactive deterring mechanisms in ABE would be interesting open problems.

2 Preliminaries

Basic Notations. If S is a set, then $s \leftarrow S$ denotes the operation of picking an element s uniformly at random from S. Let $\mathbb{N}$ denote the set of natural numbers. If $n \in \mathbb{N}$ then $[n]$ denotes the set $\{1, \ldots, n\}$. Let $z \leftarrow \mathsf{A}(x, y, \ldots)$

denote the operation of running an algorithm A with inputs $(x, y, \ldots)$ and output z. A function $f(\lambda)$ is *negligible* if for every $c > 0$ there exists a λ_c such that $f(\lambda) < 1/\lambda^c$ for all $\lambda > \lambda_c$.

Robust Fingerprinting Code. A *binary fingerprinting code* [15] is a pair of algorithms $(\mathsf{Gen}, \mathsf{Trace})$, where Gen is a probabilistic algorithm taking a number n (upper bound on the number of codewords in the system), an optional number $t \in [n] = \{1, \ldots, n\}$ (upper-bound on the detected coalition size), and security parameter ϵ as input and outputs n bit-strings $\mathcal{C} = \{C_1, \ldots, C_n\}$ (called codewords), where $C_i = \mathsf{c}_1^i \ldots \mathsf{c}_\ell^i$ for $i \in [\ell]$ and a tracing key tk. Trace is a deterministic algorithm inputting the tracing key tk and a "pirate" codeword C^*, and outputting a subset $\mathcal{U}_{acc} \subseteq [n]$ of accused users. A code is called *bias-based* [1] if each codeword $C_j = \mathsf{c}_1^j \ldots \mathsf{c}_\ell^j$ is sampled according to a vector of biases $\langle p_1, \ldots, p_\ell \rangle$, where $\forall j \in [n], \forall i \in [\ell], \Pr[\mathsf{c}_i^j = 1] = p_i$, and $p_i \in [0, 1]$.

A fingerprinting code is called $t-$*collusion resistant* (*fully collusion resistant* if $t = n$) if for any adversary $\mathcal{A}$ who corrupts up to t users (whose indices form a set $\mathcal{U}_{cor} \subset \{1, \cdots, N\}$), and outputs a pirate codeword $C^* = \mathsf{c}_1^* \ldots \mathsf{c}_n^*$ (which satisfies the marking assumption, i.e., for each $i \in [\ell], \mathsf{c}_i^* = \mathsf{c}_i^j$ for some $j \in \mathcal{U}_{cor}$),

$$\Pr[\mathcal{U}_{acc} = \emptyset \text{ or } \mathcal{U}_{acc} \not\subseteq \mathcal{U}_{cor} : \mathcal{U}_{acc} \leftarrow \mathsf{Trace}(tk, C^*)] \leq \epsilon$$

This characterizes that the probability that no users are accused or an innocent user is accused is bounded by ε.

A fingerprinting code is $\delta-$*robust* if the pirate code is further allowed to contain the symbol of '?' (not more than $\delta\ell$, where ℓ is the code length) without violating the marking assumption: now for each $i \in [\ell]$, either$\mathsf{c}_i^* = \mathsf{c}_i^j$ for some $j \in \mathcal{U}_{cor}$, or $\mathsf{c}_i^* = $ '?'.

We also recall the Tardos code [30] $F_{nt\epsilon}$ here, it has length $n = 100t^2k$, with $k = \log \frac{1}{\epsilon}$. The Gen algorithm generates a codeword as follows. For each segment index $j \in [\ell]$, it chooses a bias $p_j \in [0, 1]$ according to a distribution μ (see [30] for the definition of μ). Each bias satisfies $\frac{1}{300t} \leq p_j \leq 1 - \frac{1}{300t}$, where t is the collusion size. For each codeword $C = \mathsf{c}_1 \ldots \mathsf{c}_\ell$ outputted by Gen, $\Pr[\mathsf{c}_j = 1] = p_j$, and $\Pr[\mathsf{c}_j = 0] = 1 - p_j$ for all $j \in [\ell]$. Regarding security, there is a Trace algorithm such that, for any coalition of size at most t, with probability at least $1 - \epsilon^{t/4}$ accuses a member of the coalition, while any non-member is accused with probability at most ϵ. Note that Tardos code can be made *robust* if we extend the code length (see [3] for details).

Fingerprinted Data Transfer. Our accountable authority ABE scheme will rely on a more advanced abstraction – fingerprinted data transfer protocol – that was defined in [14]. A fingerprinted data transfer (FDT) (corresponding to a bias-based binary fingerprinting code) involves two parties, a sender S and a receiver R. The sender inputs two biases $p_0, p_1 \in [0, 1]$, four messages $(m_0^0, m_0^1), (m_1^0, m_1^1)$, and a bit $c \in \{0, 1\}$; At the end of the protocol, R outputs $\{m_i^{b_i}\}$ for $i, b_i \in \{0, 1\}$ such that $\Pr[b_i = 1] = p_i$; while S outputs b_c. The fingerprinted data transfer functionality can be expressed as:

$$\mathsf{FDT}[\perp, ((p_0, p_1), (m_0^0, m_0^1, m_1^0, m_1^1), c)] = [(m_0^{b_0}, m_1^{b_1}), b_c], \text{where } \Pr[b_i = 1] = p_i.$$

The security of a fingerprinted data transfer protocol follows the standard simulation based paradigm, for details we refer to Appendix A.1.

3 Generic Construction of Traceable Attribute Based Encryption

In this section, we will discuss our generic construction of traceable CP-ABE scheme. First we present the formal definitions.

3.1 Definition and Security Models

Traceable CP-ABE. Concretely, a traceable CP-ABE scheme consists of the following five algorithms:

$\mathsf{Setup}(n, \lambda)$: The setup algorithm takes as input the number of users n in the system and a security parameter λ, outputs a master secret key msk, a potential tracing key tk and the public parameters mpk.

$\mathsf{KeyGen}(mpk, msk, i, S_i)$: The key generation algorithm takes as input the public parameter mpk, the master secret key msk and a set of attributes S_i. It outputs a private decryption key sk_{i,S_i}, which is assigned and identified by a unique index $i \in \{1, \dots, n\}$.

$\mathsf{Enc}(mpk, m, \mathbb{A})$: The encryption algorithm takes as input the public parameters mpk, a message m and a decryption policy that is represented by an access structure $\mathbb{A}$. It outputs a ciphertext c.

$\mathsf{Dec}(sk_{i,S_i}, c)$: The decryption algorithm m takes as input the public parameters mpk, a private decryption key sk_{i,S_i} and a ciphertext c. It outputs a message m or $\perp$.

$\mathsf{Trace}^{\mathrm{D}_S}(mpk, tk, S)$ The tracing algorithm takes as input the public parameters mpk, the tracing key tk, and has black-box access to a δ-useful pirate decoder D_S[2] for a set of attributes S. It outputs an index set $I \subseteq \{1, \dots, n\}$ which identifies the set of malicious users.

Security of Traceable CP-ABE. The security of traceable CP-ABE is composed of the standard semantic security and traceability. For the standard semantic security, we refer to Appendix A.2, and here we only define the strong traceability. Intuitively, the goal of the tracing algorithm to identify at least one of the colluder users, and such identified traitor's attributes should be "critical" for the pirate decoder, (also at the same time, no innocent user should be accused). Consider the following *traceability* game (which could be describing either weak traceability and strong traceability):

[2] For a non-negligible δ, a pirate decoder D_S for a set of attributes S is δ-useful, i.e. for any message m and any access structure $\mathbb{A}$ which is satisfied by S, if $\Pr[\mathrm{D}_S(\mathsf{Enc}(mpk, m, \mathbb{A})) = m] \geq \delta$.

Setup: The challenger runs the Setup algorithm to generate public parameters mpk, tracing key tk, and master secret key msk. It gives mpk to the adversary $\mathcal{A}$ and keeps tk and msk to itself.

Key Query: The adversary adaptively queries the challenger for secret keys corresponding to sets of attributes $S_1, \ldots, S_q$ for users with indices $k_1, \ldots, k_q$. In response, the challenger runs the key generation algorithm and gives the corresponding secret key sk_{k_i,S_i} to the adversary for $1 \leq i \leq q$.

Output: $\mathcal{A}$ outputs a δ-useful pirate decoder D_S for an attributes set S.

Let $C = \{k_i | 1 \leq i \leq q\}$ be the indices of the users corrupted by the adversary and I is the indices of the identified traitors, i.e. the output of $\mathsf{Trace}^{\mathrm{D}_S}(mpk, tk, S)$. The adversary $\mathcal{A}$ wins the strong traceability game if: (1) $I = \emptyset$, i.e., no one is accused; (2) or $I \not\subseteq C$, i.e., an innocent user is accused; (3) or none of the identified traitors' attributes set includes S as a subset. The meaning of the third condition characterizes that the identified traitors have to contribute to the pirate decoder their actual functional key according to their attributes.

The advantage of an adversary in the game is defined as the probability that $\mathcal{A}$ wins the strong traceability game, where the probability is taken over the random bits used by the challenger and the adversary.

Definition 1. *A traceable CP-ABE scheme is strongly traceable if all polynomial time adversaries have at most negligible advantage in the above game.*

Note that the adversary $\mathcal{A}$ wins the weak traceability game if we only require the adversary the first two conditions. With only such a weaker requirement, it is possible that the identified traitor does not really have the decryption capability as the decoder. Consider the following trivial generic solution: we run an ABE scheme and a traitor tracing scheme in parallel, the encryption algorithm will first split the message m into $m_1 \oplus m_2$ for a randomly chosen m_1, and encrypts m_1 using ABE scheme and m_2 using the encryption algorithm of the traitor tracing scheme. Such trivial construction can already achieve the weak traceability to identify one of the corrupted users due to the property of traitor tracing scheme. However, as these two systems are not tightly bound together, it cannot satisfy the strong traceability: User i who has the attribute S_i and user j who has the attribute S_j can collude to produce a pirate decoder that has attribute S_i, where user i contributes his partial keys of the ABE system and user j contributes his partial keys of the traitor tracing system. In this way, user j will always be identified as a traitor even if he does not have the attributes of the pirate decoder at all, i.e. attribute S_i.

3.2 A Generic Construction from Tardos Codes

Basic Intuition. The above trivial solution shows that the traitor tracing system has to be embedded into the ABE system. While it might be feasible to be based on concrete algebraic structure, from the first look, it is not clear how we can have a generic construction as ABE itself does not offer traceability. We observe that instead of considering combing a traitor tracing scheme with an

ABE, we may go to a lower level to identify some combinatoric objects that could be useful: (1) It enables identifying source with collusion resistance; (2) It can be embedded to the ABE system generically. In particular, we observe that fingerprinting codes do offer such properties simultaneously.

In more detail, in a binary fingerprinting code, everyone is assigned with a bit-string as the codeword. A collusion of corrupted users can pool their codewords together to produce a pirate code (only restricted by the marking assumption, see Sect. 2 above for details). There is a tracing algorithm that can identify a source codeword from such a pirate codeword. Moreover, such traceability can be easily built into multi-recipient encryption schemes (not only for traitor tracing). The crux here is that each codeword is just a bit string, which can be used as index for user to assign keys. In the setting of CP-ABE, we can use such string to select extra dummy attributes, and the encryption policy will expand the original policy to include such dummy attribute. During regular encryption, both ciphertext encrypting the same message regarding both dummy attributes will be present, thus the extra dummy attribute will not influence the original policy. Tracing can be facilitated by feeding two ciphertext carrying different plaintext. Based on the responses, tracer can recover a pirate codeword (that might include '?'). The robust fingerprinting code then can be used to find one corrupted codeword, thus the traitor.

We remark that the marking assumption is enforced simply by the semantic security of the encryption. More importantly, we do not run the tracing system in parallel with ABE, instead, each codeword is entangled with the attributes set, thus the identified traitor's attributes will be needed for the decoder for sure. Next, we present the formal description of the construction and analysis.

Detailed Construction. Let $(\overline{\mathsf{Setup}}, \overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ be any CP-ABE scheme, and $(\overline{\mathsf{Gen}}, \overline{\mathsf{Trace}})$ be a robust binary fingerprinting code (e.g. robust Tardos code [3]). Our generic construction of traceable CP-ABE works as follows:

$\mathsf{Setup}(n, \lambda)$: Let $\epsilon = 1/2^{\lambda}$. Run $\overline{\mathsf{Setup}}(\lambda)$ and obtain $(\overline{mpk}, \overline{msk})$; also run $\overline{\mathsf{Gen}}(n, \epsilon, \delta)$ to obtain $\{W_1, \ldots, W_n\} := \Gamma$, and tracing key tk, where $W_i \in \{0,1\}^{\ell}$, for $i = 1, \ldots, n$. Choose dummy attributes $\mathtt{Attr}^0, \mathtt{Attr}^1$, and $\{\mathtt{Attr}_i\}$ for $i = 1, \ldots, \ell$, and set

$$mpk = (\overline{mpk}, \mathtt{Attr}^0, \mathtt{Attr}^1, \mathtt{Attr}_1, \ldots, \mathtt{Attr}_{\ell}),\ msk = (\overline{msk}, \Gamma).$$

$\mathsf{KeyGen}(mpk, msk, S_i)$: Suppose user i has attribute set S_i. For $k = 1, \ldots, \ell$, let $w_k^{(i)} \in \{0,1\}$ be the k-th bit of W_i and $S_{i,k} = S_i \cup \{\mathtt{Attr}^{w_k^{(i)}}\} \cup \{\mathtt{Attr}_k\}$, run

$$\overline{sk}_{S_{i,k}} \leftarrow \overline{\mathsf{KeyGen}}(\overline{mpk}, \overline{msk}, S_{i,k})$$

Output the private key $sk_{i,S_i} = \{W_i, \overline{sk}_{S_{i,k}}\}_{k \in [\ell]}$.

$\mathsf{Enc}(mpk, m, \mathbb{A})$: Choose a random position $j \in \{1, \ldots, \ell\}$ and set $\overline{\mathbb{A}}_b = \mathbb{A} \wedge \{\mathtt{Attr}_j\} \wedge \{\mathtt{Attr}^b\}$, where $b \in \{0,1\}$. Compute

$$c_0 \leftarrow \overline{\mathsf{Enc}}(\overline{mpk}, m, \overline{\mathbb{A}}_0),\ c_1 \leftarrow \overline{\mathsf{Enc}}(\overline{mpk}, m, \overline{\mathbb{A}}_1)$$

Output the ciphertext $c = (j, c_0, c_1)$.

$\mathsf{Dec}(mpk, sk_{i,S_i}, c)$: Parse the private decryption key sk_{i,S_i} as $(W_i, \{\overline{sk}_{S_i,k}\}_{k\in[\ell]})$, and the ciphertext c as (j, c_0, c_1). If $w_j^{(i)} = 0$, output $\overline{\mathsf{Dec}}(\overline{mpk}, \overline{sk}_{S_{i,j}}, c_0)$; otherwise (i.e. $w_j^{(i)} = 1$), output $\overline{\mathsf{Dec}}(\overline{mpk}, \overline{sk}_{S_{i,j}}, c_1)$.

$\mathsf{Trace}^{\mathrm{D}_S}(mpk, tk, S)$: On input the public parameters mpk, the tracing key tk, and the claimed attribute set S of the pirate decoder, the Trace algorithm has oracle access to a δ-useful pirate decoder D_S and does the following: For each j in $\{1, \ldots, \ell\}$, proceed as follows:

1. Choose an access policy $\mathbb{A}$, it is only satisfied by the attributes set S and not satisfied by any subset of S.
2. Set $\overline{\mathbb{A}}_b = \mathbb{A} \wedge \{\mathtt{Attr}_j\} \wedge \{\mathtt{Attr}^b\}$, where $b \in \{0, 1\}$.
3. According to δ, choose proper parameter $N = O(\lambda^2 \ln \ell)$ and repeat the procedure of trying decryption for N times: Choose two random message m, compute
$$c_0 = \overline{\mathsf{Enc}}(\overline{mpk}, m, \overline{\mathbb{A}}_0), c_1 = \overline{\mathsf{Enc}}(\overline{mpk}, 0, \overline{\mathbb{A}}_1),$$
$$c'_0 = \overline{\mathsf{Enc}}(\overline{mpk}, m, \overline{\mathbb{A}}_0), c'_1 = \overline{\mathsf{Enc}}(\overline{mpk}, m, \overline{\mathbb{A}}_1).$$
Set $c = (j, c_0, c_1)$ and $c' = (j, c'_0, c'_1)$.
If $\mathrm{D}_S(c) = m_0$, set $w_j = 0$;
else if $\mathrm{D}_S(c') = m$ for more than $\sqrt{\lambda}$ times, set $w_j = 1$;
else, set $w_j =$ '?'.[3]
4. Set the pirate codeword $W^* = w_1 \ldots w_\ell$, and run the tracing algorithm of the fingerprinting code $\overline{\mathsf{Trace}}(W^*, tk)$ and output the traitor set I.

Security Analysis. First, semantic security is straightforward. The new encryption algorithm is simply run the ABE scheme twice. Furthermore, each ciphertext is encrypted using a more restricted policy. We omit the details for this property.

Next, we discuss why our construction satisfies strong traceability. First, for simplicity, let us consider the case for $\delta = 1$, i.e., the decoder works perfectly on S. Suppose for a position i, if all $w_i^{(j)} = 0$ for $j \in \mathcal{U}_{cor}$ (the corrupted users), then due to the semantic security, D_S will always output the correct decryption, as the tracing ciphertext c now looks indistinguishable from the regular ciphertext, thus we can correctly capture $w_i = 0$. Similarly, if all $w_i^{(j)} = 1$ for $j \in \mathcal{U}_{cor}$, we can see that D_S will never answer m in the first stage of tests and will always answer m in the second stage of tests, and we again correctly captures $w_i = 1$. The complex case is that there are both 0 and 1 for this position i, then the pirate decoder has to make a decision (including not responding, which yields a "?" that can be handled by a *robust* fingerprinting code). If the decoder answers m correctly, we will set $w_i = 0$; otherwise, the Trace algorithm moves to the second stage of tests. Now because c' is identically distributed as a regular ciphertext, according to correctness, D_S will answer correctly and Trace can correctly capture $w_i = 1$.

[3] The tracing idea is similar to the tracing mechanism due to Boneh, Naor [4].

To summarize above, the Trace algorithm will always return a pirate codeword that satisfies the marking assumption, i.e., for each $i \in [\ell], w_i = w_i^{(j)}$ for some $j \in \mathcal{U}_{cor}$. Then the traceability of the fingerprinting code scheme ensures $I \neq \emptyset \wedge I \subseteq \mathcal{U}_{cor}$, where I is the indices of the identified traitors, i.e. the output of $\mathsf{Trace}^{\mathrm{D}_S}(mpk, tk, S)$. Last, let us argue that there exist $j \in I$, such that $S_j \subseteq S$. Suppose D_S only uses keys whose attributes do not satisfy the policy $\mathbb{A}$, then it can never decrypt correctly due to semantic security (especially the collusion resistance of ABE itself). As our Trace algorithm takes action mostly based on a correct answer, this means the "useful" keys in the pirate decoder are all those whose attribute set includes S. To put it another way, the pirate codeword captured by the Trace algorithm is actually generated using codewords of those "useful" keys only.

An imperfect decoder can also be addressed by repetition (and also the robustness of the fingerprinting codes). As the pirate decoder D_S satisfies δ-correctness, that means for ciphertext policy that the claimed attribute set S satisfies, the decoder will answer correctly with probability at least δ. It follows that at most for $\delta \cdot n$ positions, D_S stops working, i.e., for δn many positions i, try decryption using D_S by feeding (i, c_0, c_1) do not give meaningful responses, which yields $w_i =$'?'. For other positions, D_S will function properly, and the above analysis still holds.

As the intuition is not too involved and due to space limit, we defer the complete analysis to the full version and we summarize the security as follows:

Theorem 1. *If the underlying CP-ABE* ($\overline{\mathsf{Setup}}, \overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}}$) *is semantically secure, and the fingerprint code* ($\overline{\mathsf{Gen}}, \overline{\mathsf{Trace}}$) *is δ-robust and fully collusion resistant, then above CP-ABE scheme is semantically secure and strongly traceable.*

4 Enforcing Authority Accountability

4.1 Definitions and Security Models

As we mentioned in the introduction, the main requirement of authority accountability in traceable ABE is for the following reason: suppose user i is identified from a leaked decoder, however there is also possibility that the decoder is leaked by the key generation center. This ambiguity gives malicious users excuses to evade the punishment. Similar to the concept of asymmetric traitor tracing [26] and accountable authority identity based encryption, we consider the following idea: there will be exponentially many keys per user, and the user will choose one of them obliviously. The technical challenge is to still ensure the structure of the keys (fingerprinted) to maintain the tracing capability.

The KeyGen algorithm now becomes an interactive protocol between the key generation center and each user. After a pirate decoder is noticed, the Trace algorithm will return an index set denoting the corrupted users. There will be an extra Judge protocol that is run among the key generation center, a judge and an accused user to decide whether the user is indeed responsible for the leakage of the pirate box. From above description, we see that the difference of an accountable authority ABE is at the KeyGen, Judge protocols, while the other algorithms are the same as those of traceable ABE. For detailed formal definition, we refer to Appendix A.3.

Security of Accountable Authority ABE. Again, semantic security can be easily adapted from standard definitions. Here we focus more on the security regarding traceability. The first one is the same as traceable ABE, at least one malicious user should be identified as in traceable ABE and further accused by the judge. The challenge in this new setting is that the corrupted users may try to arrange in a way that the result of Trace and Judge to be *inconsistent*. It is easy to see that the traceability in this setting simply adds one more requirement that the Judge protocol should at least accuse one user from the Trace output (actually we can achieve a much stronger requirement that the Judge will accuse all malicious traitors identified by Trace algorithm). We refer formal definition of traceability in Appendix A.3.

The second property is that innocent user cannot be framed by a key generation center, in this way, an accused user will have no excuse to deny. Consider the following non-framing game:

Setup: The adversary $\mathcal{A}$ plays the role of a malicious key generation center, generates mpk, msk and sends mpk to the challenger $\mathcal{C}$.

Key Generation: The adversary and the challenger engage in the KeyGen protocol to generate secret keys for all users. In particular, $\mathcal{A}$ selects attribute sets $S_1, \ldots, S_n$ and generate secret keys for those attribute sets. The challenger will receive secret keys $sk_{S_1}, \ldots, sk_{S_n}$, and the adversary will receive the tracing key tk.

Output: The adversary outputs a decryption box D_S for an attributes set S.

Let I be the indices of the identified traitors, i.e. the output of $\mathsf{Trace}^{\mathrm{D}_S}(mpk, tk, S)$, and I' will be the confirmed traitor indices after the Judge protocol. The adversary $\mathcal{A}$ wins the non-framing game if $I' \neq \emptyset$.

Definition 2. *An accountable authority CP-ABE scheme is non-framable if all polynomial time adversaries have at most negligible advantage in the above game.*

Remark 1. Previous work [25,32] considered only white-box traceability, thus in the non-framing game, they also have to specify one single target and only allows the adversary to run KeyGen for this single user. This essentially excludes the main challenge of traceability in the multi-recipient encryption—to defend against collusion. What's worse, this restricts adversary's power too much. As a malicious key generation center, she can obviously output the pirate decoder after issuing keys to multiple, even to all users in the system. Instead, our model removes all those restrictions and tries to capture more realistic scenarios. We also remark that we did not consider here to allow the adversary to issue decryption queries after the key generation phase [11]. We leave this as an open problem.

4.2 Generic Construction of Accountable Authority CP-ABE

Basic Intuition. As illustrated above, the basic idea is that each user will be corresponding to exponentially many secret keys, and the user will choose one of them obliviously. However, note that in our generic construction of traceable ABE, secret key of each user is with special structure and selected according to a fingerprinting code. Suppose we extend the length of the fingerprinting code, then this dummy part can correspond to many keys for one user, and this part could be oblivious to the key generation center. The major technical challenge is to achieve traceability and non-framing property simultaneously. We now draw support from the idea of an asymmetric fingerprinting.

Let us recall the main properties and building blocks of an asymmetric fingerprinting. Suppose we are using the famous Tardos codes [30], which is a bias based codes. In the asymmetric setting [14], the length of the codeword is doubled. The basic requirements are that the authority is only aware of half of the codeword, and the user is not aware of where exactly are the locations that the authority knows the corresponding codeword bits. To facilitate such a goal, a fingerprinted data transfer protocol for the bias based codes was designed [14]. After the protocol, the user will obtain a codeword (or corrected fingerprinted data) with length 2ℓ, and each bit (or the corresponding data) will be distributed according to the bias. And the authority will obtain half of the codeword obliviously according to his choice of locations. Following the security analysis of [14], we can run the original tracing algorithm of Tardos codes to identify traitors. While the judge, using the other half of the codeword, will confirm the accusation. One note we would like to emphasize is that in order to ensure the consistency during the revealing phase to the judge, each party should store the transcripts from the other, and force the other party to open correctly if a judge needs to get involved. We refer detailed protocol to [14].

Now let us look at how to upgrade our traceable ABE to support authority accountability. The key generation center first prepares the corresponding 2ℓ keys for each user (those keys are also based on the extended attribute set). Then using the biases of Tardos codes and the keys as data, the key generation center and the user execute a fingerprinted data transfer protocol as the KeyGen protocol. When a pirate decoder is noticed, the authority will run the tracing algorithm of Tardos codes using the half-codes and the bias as the tracing key. This will yield a set of colluders. If a user i claimed non-guilty, the Judge protocol will be initiated. The idea is to mimic the judge in the (asymmetric) Tardos codes setting. The user and the authority has to supply the judge with the corresponding fingerprinted data transfer protocol transcripts. The judge checks the validity of the fingerprinted data transfer protocol transcripts and uses the other half of the codeword to confirm the accusation.

Detailed Construction. Let $(\overline{\mathsf{Setup}}, \overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ be a CP-ABE system. We also use a fingerprinted data transfer system FDT regarding robust Tardos code as a major building block. Our generic construction of accountable authority CP-ABE works as follows:

$\mathsf{Setup}(\lambda)$: It first runs $\overline{\mathsf{Setup}}(\lambda)$ to obtain $(\overline{mpk}, \overline{msk})$. Let ℓ be the code length of robust Tardos code. For each $j \in [2\ell]$, choose bias $p_j \in [0,1]$ according to the distribution defined by Tardos code. Pick a bitstrings $v \in \{0,1\}^\ell$ uniformly at random, and choose dummy attributes $\mathtt{Attr}^0, \mathtt{Attr}^1, \mathtt{Attr}_i$ for $i = 1, \dots, 2\ell$. Initialize a set $\mathcal{W} = \emptyset$, and set the tracing key $tk = (\{p_j\}_{j \in [2\ell]}, v, \mathcal{W})$. Set

$$mpk = (\overline{mpk}, \mathtt{Attr}^0, \mathtt{Attr}^1, \{\mathtt{Attr}_i\}_{i \in [2\ell]}),\ msk = (\overline{msk}, \{p_j\}_{j \in [2\ell]}, v)$$

and output the public parameters mpk.

$\mathsf{KeyGen}(\cdot)$: This is a protocol between the key generation center and the user. The key generation center inputs mpk, msk, S_k, and the user inputs mpk, S_k, where S_k is an attribute set.

The key generation center parses the master secret key msk as $(\overline{msk}, \{p_j\}_{j \in [2\ell]}, v)$, and write v as $v_1 \dots v_\ell$ where $v_i \in \{0,1\}$ is the i-th bit of v for $i \in [\ell]$. For each $i \in [2\ell]$ and $b \in \{0,1\}$, let $S^b_{k,i} = S_k \cup \{\mathtt{Attr}_i\} \cup \{\mathtt{Attr}^b\}$, and run

$$\overline{sk}^b_{S_{k,i}} \leftarrow \overline{\mathsf{KeyGen}}(\overline{mpk}, \overline{msk}, S^b_{k,i})$$

Then, for each $i \in [\ell]$, the authority and the user runs the fingerprinted data transfer protocol (FDT), where the authority inputs two biases p_{2i-1}, p_{2i}, four messages $(\overline{sk}^0_{S_{k,2i-1}}, \overline{sk}^1_{S_{k,2i-1}}), (\overline{sk}^0_{S_{k,2i}}, \overline{sk}^1_{S_{k,2i}})$, and a bit v_i.

At the end of the protocol, the user obtains $\overline{sk}^{w_{2i-1}}_{S_{k,2i-1}}, \overline{sk}^{w_{2i}}_{S_{k,2i}}$ where $w_{2i-1}, w_{2i} \in \{0,1\}$ and $\Pr[w_{2i-1} = 1] = p_{2i-1}, \Pr[w_{2i} = 1] = p_{2i}$, and the authority obtains the bit w_{2i-1+v_i} denoted as $\bar{w}_i$. Note that the fingerprinted data transfer may already contain the necessary committing or zero-knowledge proof steps to ensure both parties to follow the protocol.

The user's private key is set as $sk_{S_k} = (w = w_1 \dots w_{2\ell}, \{\overline{sk}^{w_i}_{S_{k,i}}\}_{i \in [2\ell]})$.

The authority uses the half-codeword $\bar{w} = \bar{w}_1 \dots \bar{w}_\ell$ (which is part of the user codeword) to identify the user, and adds the codeword $\bar{w}$ to the set $\mathcal{W}$, which is used to store the half-code of all the users.

$\mathsf{Enc}(mpk, m, \mathbb{A})$: Choose a random position $j \in \{1, \dots, 2\ell\}$ and set $\overline{\mathbb{A}}_b = \mathbb{A} \wedge \{\mathtt{Attr}_j\} \wedge \{\mathtt{Attr}^b\}$, where $b \in \{0,1\}$. Compute

$$c_0 \leftarrow \overline{\mathsf{Enc}}(\overline{mpk}, m, \overline{\mathbb{A}}_0),\ c_1 \leftarrow \overline{\mathsf{Enc}}(\overline{mpk}, m, \overline{\mathbb{A}}_1)$$

Output the ciphertext $c = (j, c_0, c_1)$.

$\mathsf{Dec}(mpk, sk_{S_k}, c)$: Parse the private decryption key sk_{S_k} as $(w = w_1 \dots w_{2\ell}, \{\overline{sk}^{w_i}_{S_{k,i}}\}_{i \in [2\ell]})$, and the ciphertext c as (j, c_0, c_1). If $w_i = b$, output

$$\overline{\mathsf{Dec}}(\overline{mpk}, \overline{sk}^b_{S_{k,i}}, c_b).$$

$\mathsf{Trace}^{\mathrm{D}_S}(mpk, tk, S, \delta)$: The Trace algorithm has only oracle access to a pirate decoder D_S. Parse the master secret key msk as $(\overline{msk}, \{p_j\}_{j \in [2\ell]}, v = v_1 \dots v_\ell)$. Let $T = \{2i - 1 + v_i\}_{i \in [\ell]}$ be the subset of locations that the key generation center knows the half-code of the user. For each $j \in T$, run the Trace algorithm of our traceable ABE scheme in Sect. 3 and output a set I of traitor indices.

Judge$(\cdot)$ This is a protocol among the key generation center, an identified traitor i who does not commit guilty and the judge. The key generation center is with input $(mpk, msk, \mathrm{D}_S, tk)$, the user is with input (mpk, sk_{S_k}), and the judge is with input (mpk, D_S).

1. The user first reveals the complete codeword of her, and prove its correctness according to the FDT protocol transcript.
2. The key generation center sends the judge the set $T = \{2j - 1 + v_j\}_{j\in[\ell]}$ of locations that the key generation knows the half-code of the user, and proves its validity to the FDT protocol transcripts.
3. The judge then runs the Trace algorithm on the locations of $[2\ell] - T$ (i.e., the set $\{2j - v_j\}_{j\in[\ell]}$) via oracle access to D_S, and obtains another half pirate codeword. Then the judge runs a slightly different tracing algorithm for the underlying Tardos fingerprinting code to decide whether user i is accused for this half of the pirate codeword (see [14] for details), output 1 if yes.
4. If the judge outputs 1 in the above step, the user will be accused; otherwise, the user will not be accused.

Remark 2. During the protocols, to enforce each party to be honest, we carried zero-knowledge proofs at various steps. However, there are several simple optimizations from [14]. As it is enough to demonstrate the idea here, we omit the details and refer to [14] for optimizations. Furthermore, [14] even achieved group accusation, i.e., *all* identified traitors can be confirmed by the judge.

Security Analysis. Semantic security is straightforward as in the traceable ABE case. Now let us take a closer look at the traceability and non-framing properties.

Regarding traceability, compared with that in the traceable ABE, there are two more chances for a malicious user to evade tracing. The first is during the key generation protocol, whether the user can obtain information about keys she is not supposed to know, or reveal incorrect information about half of her codeword. It is easy to see that this cannot happen due to the (sender) security of the fingerprinted data transfer protocol. The second is whether the malicious user can cause inconsistency during the Trace and Judge phases. We note here that as we can extract both halves of the codeword out, this problem essentially reduces to the property of the underlying Tardos fingerprinting code. Fortunately, this property of Tardos code was formally demonstrated in [14]. The last is during the Judge protocol to fool the judge about her complete codeword, the soundness of the proofs in those steps ensures that this cannot happen.

Regarding non-framing property, there are only a few places that the malicious key generation center can cheat. The first is in the key generation protocol, there is more information leaked to the key generation center (KGC) than half of the codeword chosen according to the locations by KGC. This can be prevented by the security of the (receiver) security of the FDT protocol. The second is during the Judge protocol, again, the proofs are easily verifiable.

We remark that the FDT protocol satisfies the standard simulation security, thus the composition lemma [5] can be applied and we can replace such functionality as an oracle during the analysis.

With the above security intuitions, and due to page limit, we defer detailed security proof to the full version, and we summarize the security as follows:

Theorem 2. *If the underlying CP-ABE* $(\overline{\mathsf{Setup}}, \overline{\mathsf{KeyGen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ *is semantically secure, and the fingerprint code* $(\overline{\mathsf{Gen}}, \overline{\mathsf{Trace}})$ *is δ-robust and fully collusion resistant, and the fingerprinted data transfer protocol satisfies the simulation security, then above CP-ABE scheme is semantically secure, strongly traceable and non-framable.*

Acknowledgement. We are grateful to the anonymous reviewers for their helpful comments. The work of Junzuo Lai was supported by National Natural Science Foundation of China (No. 61572235), and Guangdong Natural Science Funds for Distinguished Young Scholar (No. 2015A030306045). Qiang Tang was partially supported by NSFC Fund for Oversea Chinese Scholars (No. 61728208).

A Omitted Definitions

A.1 Simulation Based Security of Fingerprinted Data Transfer

If a protocol satisfies the following properties, we say that it securely implements fingerprinted data transfer.

$\mathsf{Correctness}$: The receiver will obtain $(m_0^{b_0}, m_1^{b_1})$, satisfying that $\Pr[b_i = 1] = p_i$ for $i = 0, 1$. The sender will receive b_c with probability 1.

$\mathsf{Receiver\ Security}$: The joint distribution of sender's view and the outputs in a real the protocol can be simulated by the inputs and outputs of the sender alone together with the ideal outputs of the functionality. That is, $\forall PPT$ semi-honest sender $\mathcal{S}$, $\exists$ PPT $\mathcal{S}'$, s.t., $VIEW_{\mathcal{S}} \circ OUTPUT$ is computationally indistinguishable from $\mathcal{S}'([(p_0, p_1), (m_0^0, m_0^1, m_1^0, m_1^1), c], b_c) \circ (m_0^{b_0}, m_1^{b_1}, b_c)$.

$\mathsf{Sender\ Security}$: The joint distribution of receiver's view and the outputs in a real protocol can be simulated by the inputs and outputs of the receiver alone, together with the ideal outputs. That is, $\forall PPT$ semi-honest receiver $\mathcal{R}$, $\exists$ PPT $\mathcal{R}'$, s.t., $VIEW_{\mathcal{R}} \circ OUTPUT$ is computationally indistinguishable from $\mathcal{R}'(m_0^{b_0}, m_1^{b_1}) \circ (m_0^{b_0}, m_1^{b_1}, b_c)$. (here we assume the bits of the codeword $\{b_i\}$ are publicly recoverable from $\{m_i^{b_i}\}$.)

A.2 Semantic Security of Traceable ABE

$\mathsf{Semantic\ Security\ Game}$. The game between a challenger and an adversary proceeds as follows:

Setup. The challenger runs the Setup algorithm to generate public parameters mpk, tracing key tk and master secret key msk. It gives mpk to the adversary and keeps tk and msk to itself.

Query Phase 1. Proceeding adaptively, the adversary can repeatedly query the challenger for secret keys corresponding to sets of attributes. In response, the challenger runs the key generation algorithm and gives the corresponding secret key to the adversary.

Challenge. The adversary submits two equal length messages m_0, m_1 and a challenge access structure $\mathbb{A}^*$ such that none of the queried attributes sets in **Query Phase 1** satisfies the challenge access structure $\mathbb{A}^*$. The challenger flips a random coin $\beta \in \{0,1\}$, and runs $\mathsf{Enc}(mpk, m_\beta, \mathbb{A}^*)$ to get the challenge ciphertext c^*. The resulting c^* is given to the adversary.

Query Phase 2. The adversary continues to adaptively issue private key queries as **Query Phase 1** with the restriction that the adversary can not issue queries on sets of attributes which satisfy the access structure $\mathbb{A}^*$.

Guess. Finally, the adversary $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$. The adversary wins if $\beta' = \beta$.

The advantage of $\mathcal{A}$ in this game is defined as $|\Pr[\beta' = \beta] - \frac{1}{2}|$, where the probability is taken over the random bits used by the adversary $\mathcal{A}$ and $\mathcal{C}$.

Definition 3. *A traceable CP-ABE scheme is semantically secure if all polynomial time adversaries have at most negligible advantage in the above game.*

A.3 Accountable Authority CP-ABE

Concretely, an accountable authority CP-ABE scheme consists of the following five algorithms:

Setup. The setup algorithm takes as input a security parameter λ, outputs a master secret key msk, the tracing key tk and the public parameters mpk.

KeyGen. This is an interactive protocol between the key generation center and a user. The common input to key generation center and the user are the public parameters mpk and the attributes set S of the user. The private input to key generation center is the master secret key msk. At the end of the protocol, the user receives a private key sk_S, which is assigned and identified by a unique index.

Enc. The encryption algorithm takes as input the public parameters mpk, a message m and an access structure $\mathbb{A}$. It outputs a ciphertext c.

Dec. The decryption algorithm m takes as input the public parameters mpk, a private decryption key sk_S and a ciphertext c. It outputs a message m or $\perp$.

Trace. The tracing algorithm takes as input the public parameters mpk, the tracing key tk, and has black-box access to an δ-useful pirate decoder D_S for a set of attributes S. It outputs an index set I which identifies the set of malicious users.

Judge. This is an interactive protocol among the key generation center, a user who does not commit guilty and the judge. The common input to the key generation center, the user and the judge are the public parameters mpk and a δ-useful pirate decoder D_S. Additionally, the key generation center is with input the tracing key tk, and the user is with input his/her private key sk_S. At the end of the protocol, the judge decides whether the user is acquitted.

Traceability in Accountable Authority ABE. Consider the following (strong) traceability game:

Setup: The challenger runs the Setup algorithm to generate public parameters mpk, tracing key tk, and master secret key msk. It gives mpk to the adversary $\mathcal{A}$ and keeps tk and msk to itself.

Key Query: The adversary adaptively queries the challenger for secret keys corresponding to sets of attributes $S_1, \ldots, S_q$ for users with indices $k_1, \ldots, k_q$. In response, the challenger runs the key generation algorithm and gives the corresponding secret key to the adversary for $1 \leq i \leq q$.

Output: $\mathcal{A}$ outputs a δ-useful pirate decoder D_S for an attributes set S.

Let $C = \{k_i | 1 \leq i \leq q\}$ be the indices of the users corrupted by the adversary and I is the indices of the identified traitors, i.e. the output of $\mathsf{Trace}^{\mathrm{D}_S}(mpk, tk, S)$. The adversary $\mathcal{A}$ wins the strong traceability game if (1) $I = \emptyset$, i.e., no one is accused; (2) or $I \nsubseteq C$, i.e., an innocent user is accused; (3) or none of the identified traitors' attributes set includes S as a subset; (4) Judge algorithm does not accuse any of the member in C.

The advantage of an adversary in the game is defined as the probability that $\mathcal{A}$ wins the strong traceability game, where the probability is taken over the random bits used by the challenger and the adversary.

Definition 4. *An accountable authority CP-ABE scheme is strongly traceable if all P.P.T adversaries have at most negligible advantage in the above game.*

References

1. Amiri, E., Tardos, G.: High rate fingerprinting codes and the fingerprinting capacity. In: SODA, pp. 336–345 (2009)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
3. Boneh, D., Kiayias, A. Montgomery, H.W.: Robust fingerprinting codes: a near optimal construction. In: DRM, pp. 3–12 (2010)
4. Boneh, D., Naor, M.: Traitor tracing with constant size ciphertext. In: CCS, pp. 501–510 (2008)
5. Canetti, R.: Security and composition of multiparty cryptographic protocols. J. Cryptol. **13**(1), 143–202 (2000)
6. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 257–270. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_25
7. Dwork, C., Lotspiech, J.B., Naor, M.: Digital signets: self-enforcing protection of digital information (preliminary version). In: STOC, pp. 489–498 (1996)
8. Garg, S., Gentry, C., Halevi, S., Sahai, A., Waters, B.: Attribute-based encryption for circuits from multilinear maps. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 479–499. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_27
9. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-based encryption for circuits. In: STOC, pp. 545–554 (2013)

10. Goyal, V.: Reducing trust in the PKG in identity based cryptosystems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 430–447. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_24
11. Goyal, V., Lu, S., Sahai, A., Waters, B.: Black-box accountable authority identity-based encryption. In: ACM Conference on Computer and Communications Security, pp. 427–436 (2008)
12. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
13. Katz, J., Schröder, D.: Tracing insider attacks in the context of predicate encryption schemes. In: ACITA (2011)
14. Kiayias, A., Leonardos, N., Lipmaa, H., Pavlyk, K., Tang, Q.: Communication optimal tardos-based asymmetric fingerprinting. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 469–486. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16715-2_25
15. Kiayias, A., Pehlivanoglu, S.: Encryption for Digital Content. Advances in Information Security, vol. 52. Springer, Heidelberg (2010). https://doi.org/10.1007/978-1-4419-0044-9
16. Kiayias, A., Tang, Q.: How to keep a secret: leakage deterring public-key cryptosystems. In: ACM CCS 2013, pp. 943–954 (2013)
17. Kiayias, A., Tang, Q.: Traitor deterring schemes: using bitcoin as collateral for digital contents. In: ACM CCS 2015, pp. 231–242 (2015)
18. Kiayias, A., Tang, Q.: Making ***any*** identity-based encryption accountable, efficiently. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 326–346. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24174-6_17
19. Lai, J., Deng, R.H., Zhao, Y., Weng, J.: Accountable authority identity-based encryption with public traceability. In: Dawson, E. (ed.) CT-RSA 2013. LNCS, vol. 7779, pp. 326–342. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36095-4_21
20. Li, J., Huang, Q., Chen, X., Chow, S.S.M., Wong, D.S., Xie, D.: Multi-authority ciphertext-policy attribute-based encryption with accountability. In: ASIACCS, pp. 386–390 (2011)
21. Libert, B., Vergnaud, D.: Towards black-box accountable authority IBE with short ciphertexts and private keys. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 235–255. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00468-1_14
22. Liu, Z., Cao, Z., Wong, D.S.: Blackbox traceable CP-ABE: how to catch people leaking their keys by selling decryption devices on eBay. In: ACM Conference on Computer and Communications Security, pp. 475–486 (2013)
23. Liu, Z., Cao, Z., Wong, D.S.: White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures. IEEE Trans. Inf. Forensics Secur. **8**(1), 76–88 (2013)
24. Ning, J., Cao, Z., Dong, X., Wei, L., Lin, X.: Large universe ciphertext-policy attribute-based encryption with white-box traceability. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8713, pp. 55–72. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11212-1_4

25. Ning, J., Dong, X., Cao, Z., Wei, L.: Accountable authority ciphertext-policy attribute-based encryption with white-box traceability and public auditing in the cloud. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9327, pp. 270–289. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24177-7_14
26. Pfitzmann, B., Schunter, M.: Asymmetric fingerprinting. In: Maurer, U. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 84–95. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-68339-9_8
27. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: CCS, pp. 463–474 (2013)
28. Sahai, A., Seyalioglu, H.: Fully secure accountable-authority identity-based encryption. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 296–316. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_19
29. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
30. Tardos, G.: Optimal probabilistic fingerprint codes. J. ACM **55**(2), 10:1–10:24 (2008)
31. Yuen, T.H., Chow, S.S., Zhang, C., Yiu, S.M.: Exponent-inversion signatures and IBE under static assumptions. Cryptology ePrint Archive, Report 2014/311 (2014). http://eprint.iacr.org/
32. Zhang, Y., Li, J., Zheng, D., Chen, X., Li, H.: Accountable large-universe attribute-based encryption supporting any monotone access structures. In: Liu, J.K.K., Steinfeld, R. (eds.) ACISP 2016. LNCS, vol. 9722, pp. 509–524. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40253-6_31