Traceable Inner Product Functional Encryption

Xuan Thanh Do, Duong Hieu Phan, David Pointcheval

Traceable Inner Product Functional Encryption

Xuan Thanh Do[1,2], Duong Hieu Phan[2], and David Pointcheval[3,4]

[1] Department of Mathematics, Vietnam National University, Hanoi, Vietnam
[2] XLIM, University of Limoges, CNRS, France
{xuan-thanh.do,duong-hieu.phan}@unilim.fr
[3] DIENS, École normale supérieure, CNRS, PSL University, Paris, France
[4] INRIA, Paris, France
david.pointcheval@ens.fr

Abstract. Functional Encryption (FE) has been widely studied in the last decade, as it provides a very useful tool for restricted access to sensitive data: from a ciphertext, it allows specific users to learn a function of the underlying plaintext. In practice, many users may be interested in the same function on the data, say the mean value of the inputs, for example. The conventional definition of FE associates each function to a secret *decryption functional key* and therefore all the users get the same secret key for the same function. This induces an important problem: if one of these users (called a *traitor*) leaks or sells the decryption functional key to be included in a *pirate* decryption tool, then there is no way to trace back its identity. Our objective is to solve this issue by introducing a new primitive, called *Traceable Functional Encryption*: the functional decryption key will not only be specific to a function, but to a user too, in such a way that if some users collude to produce a pirate decoder that successfully evaluates a function on the plaintext, from the ciphertext only, one can trace back at least one of them.
We propose a concrete solution for Inner Product Functional Encryption (IPFE). We first remark that the ElGamal-based IPFE from Abdalla *et. al.* in PKC '15 shares many similarities with the Boneh-Franklin traitor tracing from CRYPTO '99. Then, we can combine these two schemes in a very efficient way, with the help of pairings, to obtain a Traceable IPFE with black-box confirmation.

Keywords: Functional Encryption, IPFE, Traceability.

1 Introduction

Public Key Encryption (PKE) enables people to securely communicate and share sensitive data to others over public channels. Functional Encryption (FE) [SW05, BSW11], proposed by Boneh, Sahai and Waters, overcomes some limitations of PKE. It allows recipients to recover encrypted data in a more fine grained manner. Instead of revealing all-or-nothing of the original encrypted data as in PKE, recipients can get the evaluation of (statistical) functions on the data. As the function can contain an access control that checks some relation between the identity in the functional decryption key and the authorized identity in the plaintext, this primitive generalizes Identity Based Encryption (IBE) and Attribute Based Encryption (ABE), and actually received a large interest from the community. However, there is still no efficient construction of functional encryption for general functions. Currently, there are only simple and effective constructions for linear and quadratic functions [ABDP15, ALS16, BCFG17].

In many practical applications, it is common that people only care about several specific functions on the data, for example the mean value of the data. Allowing many people to get access to the same function, with possible malicious users, has not been really covered by the previous works: the functional decryption key is derived from the function and the master secret key, but independently of the user. Therefore, all the users are given the same key, and if this key is leaked, no one can identify the origin of the leakage. The tracing problem becomes critical for this situation. We define a new primitive, called Traceable Functional Encryption (TFE).

Traitor tracing is a mechanism enabling an authority or an arbitrary party (who is a delegated party in the system to perform tracing tasks) can identify malicious users (traitors) who possibly colluded to produce a pirate decoder that behaves the same as a normal decryption. The very first traitor tracing scheme has been introduced by Chor, Fiat and Naor [CFN94] and made use of combinatorial tools. The first algebraic traitor tracing scheme has been introduced by Boneh and Franklin [BF99], and is the basis for many subsequent schemes. A large number of

schemes, in pairing-based setting or in lattice-based setting, have been introduced, we list a few of them [KY02, CPP05, BSW06, FNP07, LPSS14, BZ14, NWZ16, ABP+17, GKW18, CVW+18]. A taxonomy of the traitor tracing schemes can be found in [ABP+17, GKW18]. The classical tracing notion requires that the pirate decoder is able to decrypt random messages for being traced. In [GKW18], it is shown that there is a flaw in some tracing systems with this notion and a fix is proposed with a stronger notion which only requires the decoder to distinguish two messages of its choice. This is a very strong notion and we will consider it in this work.

Concerning advanced primitives, traceability in IBE [ADML+07, Goy07, AHL+08, PT11] and in ABE [NCD+14, LCW13, LW15a, LW15b] have been considered. Achieving traceability is usually very expensive. Adding traitor tracing to public-key encryption indeed requires a very high extra cost: even in the bounded model, the cost grows proportionally with the number of traitors. Interestingly, in the case of inner-product functional encryption, we can hope for a better deal. Indeed, in IPFE, as the number of corrupted keys is anyway bounded by the dimension of the plaintext vector and the ciphertext size is linear in this dimension, we can hope that adding traceability does not need such a huge extra cost. This is also what we achieve in this paper: adding certain level of traceability for inner-product functional encryption does not cost much. We achieve this in the discrete logarithm setting where we can note a clear similarity in the design of the first inner-product functional encryption scheme [ABDP15] and the Boneh-Franklin traitor tracing [BF99] and thus can combine them into a traceable inner-product functional encryption. It leaves as an open problem to get any level of traceability in other settings of inner-product functional encryption.

We eventually provide a construction for traceable inner-product functional encryption with black-box confirmation. Our construction is semantically secure in a more general setting than in IPFE as the adversary can choose both identity and function to query the corresponding functional secret key. Concerning traceability, it achieves *one-target tracing*: an adversary $\mathcal{A}$ is allowed to ask secret keys for one target function only, but many identities, and then produces a pirate decoder for this function. This is a basic step to be able to achieve higher level of traceability. Note that this is the security level we manage to prove, but this is still an open problem to prove it secure when the adversary can ask keys for several functions. At least, we did not find any attack in this stronger setting either.

This notion captures already useful real-life applications: suppose a group of users possesses decryption keys for the average functionality and leaks them to the pirate, if the pirate can produce a new decoder for this average function then one can trace one of the traitors. One might worry that a pirate decoder outputting an altered function (say $2F(x)$ instead of $F(x)$) might not be traced. However, as far as the target function can be computed (from public information) from the outputted function of the pirate decoder, then the traitors can still be traced. In fact, if the target function is $F^\star$ and if the pirate can output a decoder D that computes $F = 2F^\star$ then the tracer can still consider as if the decoder would output $F^\star$ because anyone can compute $F^\star$ from F. More formally, one can define a new decoder $D^\star$ for $F^\star$ (computable from F) from D for F and do tracing on $D^\star$ instead of D. We also notice that one-target tracing defeats cloned pirate decoders. In fact, the most popular way in practice to produce a pirate decoder is to clone a legitimate one with its secret key. By using the existing IPFE, one cannot trace a cloned decoder as the functional secret key does not depend on the identities of users. With one-target tracing, one can trace back the identity of the users who participated for the cloned decoder. This is indeed covered by the case that the adversary makes many queries but for one target function only, as the same function is implemented in the various decoders but for different identities. Eventually, in the theoretical sense, one-target tracing for IPFE is a stronger model than a bounded traitor tracing. Indeed if we fix a function $(1, x_2, \ldots, x_k)$, give secret keys for this function to the users, and then send the message $(m, 0, 0, ..., 0)$, then legitimates users can decrypt to the message m and the one-target tracing corresponds to a classical traitor tracing.

Our Technique. We exploit the similarities between the Boneh and Franklin's traitor tracing scheme [BF99] and the Abdalla *et. al.*'s IPFE scheme [ABDP15] to integrate the Boneh-Franklin tracing technique into the IPFE scheme of Abdalla *et. al.* [ABDP15] which allows in particular to personalize functional decryption keys. Interestingly, our method of personalizing keys and adding traceability does not need a huge extra cost as it is usually required for others primitives such as broadcast encryption.

We first informally recall the main ingredients of the IPFE of Abdalla *et. al.* [ABDP15], that encrypts a plaintext vector $\boldsymbol{y} = (y_1, \ldots, y_k)$ as follows: the master secret key $\mathsf{MSK} = \boldsymbol{s} = (s_1, \ldots, s_k)$ and the public key $\mathsf{PK} = \left(\mathbb{G}, \left(h_i = g^{s_i}\right)_{i\in[k]}\right)$ respectively allow to generate functional decryption keys and ciphertexts:

$$\mathsf{sk}_{\boldsymbol{x}} = \langle \boldsymbol{s}, \boldsymbol{x}\rangle = \sum_{i\in[k]} s_i \cdot x_i, \qquad \mathsf{CT}_{\boldsymbol{y}} = \left(g^r, \left(h_i^r \cdot g^{y_i}\right)_{i\in[k]}\right).$$

Here, we are working in a cyclic $\mathbb{G}$ of prime order q, with a generator g. The master secret key MSK is a vector $\boldsymbol{s}$ with components s_i are taken from $\mathbb{Z}_q$. The public key PK consists of k group elements h_i. The vector $\boldsymbol{x} = (x_1, \ldots, x_k)$ with components x_i is taken from $\mathbb{Z}_q$ is used to extract a functional decryption key $\mathsf{sk}_{\boldsymbol{x}}$. A ciphertext, which is generated for a plaintext $\boldsymbol{y}$, denoted by $\mathsf{CT}_{\boldsymbol{y}}$. The $\mathsf{Decrypt}$ algorithm computes

$$\prod_{i\in[k]} \left(h_i^r \cdot g^{y_i}\right)^{x_i} \times \left(g^r\right)^{-\mathsf{sk}_{\boldsymbol{x}}} = \frac{g^{\langle \boldsymbol{s},\boldsymbol{x}\rangle r} \cdot g^{\langle \boldsymbol{x},\boldsymbol{y}\rangle}}{g^{\langle \boldsymbol{s},\boldsymbol{x}\rangle r}} = g^{\langle \boldsymbol{x},\boldsymbol{y}\rangle}$$

and gets $\langle \boldsymbol{x}, \boldsymbol{y}\rangle$, which is supposed to be relatively small, to allow the computation of the discrete logarithm.

For the mean value, the vector $\boldsymbol{x}$ is $(1, \ldots, 1)$. If many users are interested in the mean value then they all get the same functional decryption key $\mathsf{sk}_{\boldsymbol{x}}$ and there will be no way to trace the source of the leakage if this secret key is used somewhere. In order to personalize functional decryption keys for each vector $\boldsymbol{x}$, we have got inspired from the seminal technique of Boneh-Franklin: we associate to each user a representation of $g^{\langle \boldsymbol{s},\boldsymbol{x}\rangle}$ in the basis of $\left(b_i = g^{t_i}\right)_{i\in[k]}$, with t_i is taken from $\mathbb{Z}_q$. Therefore, by adding b_i^r in the ciphertext, each user can compute $g^{\langle \boldsymbol{s},\boldsymbol{x}\rangle r}$ as above and the decryption works in the same manner. Concretely, each user ID is associated to a public codeword $\boldsymbol{\theta}_{\mathsf{ID}} = (\theta_1, \ldots, \theta_k)$ and then, the personal secret key will be simply set to: $\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}} = \langle \boldsymbol{s}, \boldsymbol{x}\rangle / \langle \boldsymbol{t}, \boldsymbol{\theta}_{\mathsf{ID}}\rangle$. The master secret key MSK consists of two vectors $\boldsymbol{s} = (s_1, \ldots, s_k)$ and $\boldsymbol{t} = (t_1, \ldots, t_k)$. The public key $\mathsf{PK} = \left(\mathbb{G}, \left(b_i = g^{t_i}\right)_{i\in[k]}, \left(h_i = g^{s_i}\right)_{i\in[k]}\right)$. For each plaintext $\boldsymbol{y}$, the ciphertext is

$$\mathsf{CT}_{\boldsymbol{y}} = \left(\left(b_i^r\right)_{i\in[k]}, \left(h_i^r \cdot g^{y_i}\right)_{i\in[k]}\right).$$

The $\mathsf{Decrypt}$ algorithm then outputs

$$\prod_{i\in[k]} \left(h_i^r \cdot g^{y_i}\right)^{x_i} \times \prod_{i\in[k]} \left(b_i^r\right)^{-\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}}\theta_i} = \frac{g^{\langle \boldsymbol{s},\boldsymbol{x}\rangle r} \cdot g^{\langle \boldsymbol{x},\boldsymbol{y}\rangle}}{g^{\langle \boldsymbol{s},\boldsymbol{x}\rangle r}} = g^{\langle \boldsymbol{x},\boldsymbol{y}\rangle}.$$

The use of pairings. The above technique of personalizing secret keys seems to work well as in the Boneh-Franklin traitor tracing. However, there exists an issue specific to the setting of the functional encryption, that goes beyond the framework of Boneh-Franklin traitor tracing. Suppose that we are considering a scheme for two users with identities ID_1 and ID_2. The first user queries the secret keys corresponding to vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ and gets $\mathsf{tk}_{\boldsymbol{x}_1,\mathsf{ID}_1} = \frac{\langle \boldsymbol{s},\boldsymbol{x}_1\rangle}{\langle \boldsymbol{t},\boldsymbol{\theta}_{\mathsf{ID}_1}\rangle}$ and $\mathsf{tk}_{\boldsymbol{x}_2,\mathsf{ID}_1} = \frac{\langle \boldsymbol{s},\boldsymbol{x}_2\rangle}{\langle \boldsymbol{t},\boldsymbol{\theta}_{\mathsf{ID}_1}\rangle}$. The second user only queries secret key to vector $\boldsymbol{x}_1$ and gets $\mathsf{tk}_{\boldsymbol{x}_1,\mathsf{ID}_2} = \frac{\langle \boldsymbol{s},\boldsymbol{x}_1\rangle}{\langle \boldsymbol{t},\boldsymbol{\theta}_{\mathsf{ID}_2}\rangle}$. From these three secret keys $\mathsf{tk}_{\boldsymbol{x}_1,\mathsf{ID}_1}, \mathsf{tk}_{\boldsymbol{x}_2,\mathsf{ID}_1}$ and $\mathsf{tk}_{\boldsymbol{x}_1,\mathsf{ID}_2}$, it is possible to compute the secret key

$\mathsf{tk}_{\boldsymbol{x}_2,\mathsf{ID}_2} = \frac{\mathsf{tk}_{\boldsymbol{x}_2,\mathsf{ID}_1} \cdot \mathsf{tk}_{\boldsymbol{x}_1,\mathsf{ID}_2}}{\mathsf{tk}_{\boldsymbol{x}_1,\mathsf{ID}_1}}$ for the vector $\boldsymbol{x}_2$ and identity ID_2. To avoid this attack, we will put the scalar $t_{\boldsymbol{x},\mathsf{ID}}$ in the exponent $\mathsf{sk}_{\boldsymbol{x},\mathsf{ID}} = g^{\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}}}$ and the decryption will then be performed in the target group of the pairing. The goal of the rest of the paper is to prove this modification actually leads to a secure scheme.

Enhancing the security of IPFE. It is worth noticing that, by putting the secret key in the exponent, we may enhance the security of the functional encryption. In the Abdalla *et. al.*'s scheme [ABDP15], whenever the adversary queries more than k secret keys, it can get the whole MSK by solving a system of linear equations. In our scheme, there is no way, unless breaking discrete logarithm, to get this master key as it is only put in the exponent. We will though not exploit further this advantage in this work, as we will focus on traceability.

Tracing algorithm. We rely on the classical linear tracing technique but we will adapt this technique into the functional encryption setting and with the strongest notion of pirate, namely pirate distinguisher introduced in [GKW18].

Organization. In Section 2, we will recall some classical assumptions (DDH and BDDH), required for the security of our constructions. In Section 3, we introduce a new concept: Traceable functional encryption (TFE). We then define security game of TFE against adaptively-chosen plaintext attacks and security game of the Tracing algorithm. A concrete TFE construction for inner product will be presented in Section 4. We will prove that our construction achieves selective security. Section 5 will be intended to present a tracing algorithm which achieves one-target security as stated in Theorem 13. The black-box confirmation property of the Tracing algorithm will be proven in the Lemma 11 and 12.

2 Preliminaries

We denote $[k]$ the set of integers between 1 and k. Given two vectors $\boldsymbol{x} = (x_1, \ldots, x_k)$ and $\boldsymbol{y} = (y_1, \ldots, y_k)$, where $x_i, y_i \in \mathbb{Z}_q$ for all $i \in [k]$, we define $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{k} x_i y_i$. Next we recall classical assumptions as follows.

Definition 1 (Decisional Diffie-Hellman Assumption). Given a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order q, the Decision Diffie Hellman (DDH) problem consists in distinguishing the following distributions

$$\mathcal{D}_0 = \{(g^a, g^b, g^{ab}) \mid a, b \xleftarrow{\$} \mathbb{Z}_q\} \qquad \mathcal{D}_1 = \{(g^a, g^b, g^c) \mid a, b, c \xleftarrow{\$} \mathbb{Z}_q\}.$$

The distribution $\mathcal{D}_0$ consists of Diffie-Hellman tuples whereas $\mathcal{D}_1$ consists of random tuples. Roughly speaking, the DDH problem consists in distinguishing DH tuples from random tuples. The DDH assumption states that the two above distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are indistinguishable.

Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be multiplicatively written groups of prime order q, and let g_1, g_2 be generators of $\mathbb{G}_1, \mathbb{G}_2$, respectively. We write 1_T to denote the unit element of $\mathbb{G}_T$. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a function sending two elements from $\mathbb{G}_1$ and $\mathbb{G}_2$ into the group $\mathbb{G}_T$. We say that the tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ is an asymmetric bilinear group if the following properties hold:

- **Bilinearity**: for all $h_1 \in \mathbb{G}_1, h_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_q^*$, we have $e(h_1^a, h_2^b) = e(h_1, h_2)^{ab}$.
- **Non-degeneracy**: $e(g_1, g_2) \neq 1_T$.
- The function e can be efficiently computed.

Definition 2 (Bilinear Decisional Diffie-Hellman Assumption).
Given an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$, the Bilinear Decisional Diffie-Hellman

(BDDH) problem consists in distinguishing the following distributions, for generators g_1 and g_2

$$\mathcal{D}_0 = \left\{ \left(g_1^a, g_1^b, g_2^a, g_2^c, e\left(g_1, g_2\right)^{abc} \right) | a, b, c \xleftarrow{\$} \mathbb{Z}_q \right\}$$
$$\mathcal{D}_1 = \left\{ \left(g_1^a, g_1^b, g_2^a, g_2^c, e\left(g_1, g_2\right)^{z} \right) | a, b, c, z \xleftarrow{\$} \mathbb{Z}_q \right\}.$$

The BDDH assumption states that no PPT adversary can distinguish $\mathcal{D}_0$ and $\mathcal{D}_1$ with non negligible advantage.

Lemma 3 (Two-tailed Chernoff Bound). *Let $X_1, X_2, \ldots, X_n$ be independent Poisson trials (yes/no experiments) with success probabilities $p_1, p_2, \ldots, p_n$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \sum_{i=1}^{n} p_i$. For $0 < \delta < 1$, we have*

$$\Pr[|X - \mu| \geq \delta\mu] \leq 2e^{-\mu\delta^2/3}.$$

3 Traceable Functional Encryption

We begin by describing the syntactic definition of traceable functional encryption (TFE) for circuits. A functionality (circuit) $F \in \mathcal{F}_\lambda$ describes the function of a plaintext that can be derived from the ciphertext. More precisely, a functionality is defined as follows.

Definition 4. Let $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda\in\mathbb{N}}$ denote ensembles where each $\mathcal{Y}_\lambda$ and $\mathcal{S}_\lambda$ is a finite set. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ denotes an ensemble where each $\mathcal{F}_\lambda$ is a finite collection of circuits, and each circuit $F \in \mathcal{F}_\lambda$ takes as input a message $y \in \mathcal{Y}_\lambda$ and outputs $F(y) \in \mathcal{S}_\lambda$.

Definition 5. A traceable functional encryption scheme $\mathcal{T} - \mathcal{FE}$ for an ensemble $\mathcal{F}$ consists of five algorithms $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Tracing})$ defined as follows:

$\mathsf{Setup}(1^\lambda)$**:** Takes as input a security parameter λ and outputs a master key pair $(\mathsf{PK}, \mathsf{MSK})$.

$\mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, F)$**:** Given an identity ID of a user, a circuit $F \in \mathcal{F}_\lambda$ and the master secret key MSK, this algorithm outputs an individual functional secret key $\mathsf{sk}_{F,\mathsf{ID}}$.

$\mathsf{Encrypt}(\mathsf{PK}, y)$**:** Takes as input the public key PK and a message $y \in \mathcal{Y}_\lambda$, this randomized algorithm outputs a ciphertext CT.

$\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{F,\mathsf{ID}}, \mathsf{CT})$**:** Given the public key PK, a secret key $\mathsf{sk}_{F,\mathsf{ID}}$ and a ciphertext CT, this algorithm outputs $F(y) \in \mathcal{S}_\lambda$, if CT encrypts y, or an invalid symbol $\perp$.

$\mathsf{Tracing}^{\mathcal{D}_F}(\mathsf{MSK}, F, \mu(.), y^0, y^1)$**:** The tracing algorithm takes as input the master secret key MSK, a circuit $F \in \mathcal{F}_\lambda$, two messages $y^0, y^1 \in \mathcal{Y}_\lambda$ which are obtained from $\mathcal{D}_F$ and a function $\mu(.)$ representing the probability that the decoder can distinguish between the ciphertexts of y^0 and of y^1. The algorithm interacts with a confiscated pirate decoder $\mathcal{D}_F$, as a black-box, and outputs an identity or an invalid symbol $\perp$.

For correctness, we require that for all $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$, all $y \in \mathcal{Y}_\lambda$, each $F \in \mathcal{F}_\lambda$ and all identities ID, $\mathsf{sk}_{F,\mathsf{ID}} \leftarrow \mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, F)$, if $\mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{PK}, y)$, then one should get $\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{F,\mathsf{ID}}, \mathsf{CT}) = F(y)$, with overwhelming probability.

Requirement on the pirate decoder

- The classical requirement is that the pirate decoder $\mathcal{D}_F$ is a device that is able to decrypt successfully any normal ciphertext generated by the $\mathsf{Encrypt}$ algorithm with high probability. Yet, in another approach, the tracer is only able to interact with $\mathcal{D}_F$ through an oracle $\mathcal{O}_F^{\mathcal{D}}$ by sending a message-ciphertext pair (tracing signal) to $\mathcal{O}_F^{\mathcal{D}}$ and gets a response that is a bit indicating whether $\mathcal{D}_F$ can successfully decrypt the ciphertext into the provided message (evaluated with the function F). We say that the tracing algorithm is executing in minimal access black-box mode.

$$\mathcal{O}_F^{\mathcal{D}}(\mathsf{CT}, y) = \begin{cases} 1 & \text{if } \mathcal{D}_F(\mathsf{CT}) = F(y) \\ 0 & \text{otherwise.} \end{cases}$$

– We consider the same setting for the pirate as in [GKW18]: of course, this is not required the pirate decoder $\mathcal{D}_F$ to output entire message (or an indicator bit as in minimal access model) nor to decrypt with high probability every ciphertexts which are taken from random messages. Instead, it is enough that the pirate decoder can distinguish the encryption of two messages y^0, y^1 which are chosen by itself (see [GKW18]): Adapted from [GKW18], we define a μ-useful Pirate Distinguisher $\mathcal{D}_F$ associated to a unique function F as below

$$\left| \mathsf{Pr} \left[\begin{array}{c} (\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(\cdot) \\ \{\mathsf{sk}_{F,i} \leftarrow \mathsf{Extract}(i, \mathsf{MSK}, F)\}_{i \in [n]} \\ \mathcal{D}_F\left(\mathsf{CT}_b\right) = b : (\mathcal{D}_F, y^0, y^1) \leftarrow \mathcal{A}(\mathsf{PK}, \{\mathsf{sk}_{F,i}\}_{i \in [t]}) \\ \text{st. } F(y^0) \neq F(y^1) \\ b \xleftarrow{\$} \{0,1\}, \mathsf{CT}_b \leftarrow \mathsf{Encrypt}(\mathsf{PK}, y^b) \end{array} \right] - \frac{1}{2} \right| \geq \mu(\lambda),$$

where the function $\mu(\cdot)$ is a non-negligible function in λ.
This very strong notion of Pirate Distinguisher has been introduced in [GKW18]. It requires the pirate distinguisher to be able to distinguish the encryption of two different messages y^0, y^1. To adapt to the functional encryption, as the goal of the pirate is to compute the function on the message, we require that the pirate distinguisher be able to distinguish the encryption of y^0, y^1 such that $F(y^0) \neq F(y^1)$.
As shown in [GKW18], this notion is stronger than the classical Pirate Decoder which is able to correctly decrypt random messages with non-negligible probability. When considering the case of functional encryption, a pirate decoder for a function F is useful if it can compute $F(y)$ from the encryption of y, for a random message y. Clearly, pirate distinguisher is also stronger than pirate decoder in this case. Indeed, one can build a distinguisher $\mathcal{D}_F$ from a decoder Dec_F: randomly choose y^0, y^1 such that $F(y^0) \neq F(y^1)$, then when receiving the challenge ciphertext CT, call Dec_F and check whether this is $F(y^0)$ or $F(y^1)$ to output the correct guess, if this is none of them, output a random guess. In this work, we will deal with this notion of pirate distinguisher which is actually the strongest notion (*i.e.*, minimal requirement) about the usefulness of pirate decoders.

Security: Indistinguishability. We consider the IND security game between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$ as follows:

Definition 6. A traceable functional encryption scheme $\mathcal{T}-\mathcal{FE}$ for an ensemble $\mathcal{F}$, $\mathcal{T}-\mathcal{FE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Tracing})$ is semantically secure under chosen-plaintext attacks (or $\mathsf{IND}-\mathsf{CPA}$ security) if no PPT adversary has non-negligible advantage in the following game:

– The challenger $\mathcal{B}$ runs $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and the public key PK is given to the adversary $\mathcal{A}$.
– The adversary adaptively makes secret key queries to the challenger. That is, the adversary $\mathcal{A}$ chooses some pairs of identities ID and functions $F \in \mathcal{F}_\lambda$. $\mathcal{A}$ sends them to $\mathcal{B}$ and then obtains $\mathsf{sk}_{F,\mathsf{ID}} \leftarrow \mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, F)$ from $\mathcal{B}$.
– The adversary $\mathcal{A}$ chooses distinct messages $y_0, y_1 \in \mathcal{Y}_\lambda$ such that $F(y_0) = F(y_1)$ for all F already asked. This restriction is required in all functional encryption to avoid trivial attacks. Whenever $\mathcal{B}$ receives the messages, it randomly picks $\beta \xleftarrow{\$} \{0,1\}$ and then transfers to $\mathcal{A}$ a ciphertext $\mathsf{CT}_\beta = \mathsf{Encrypt}(\mathsf{PK}, y_\beta)$.
– Adversary $\mathcal{A}$ continues making further decryption key queries for other pairs of identities ID and functions F, and receives $\mathsf{sk}_{F,\mathsf{ID}}$ from $\mathcal{B}$. Again, it is also required that $F(y_0) = F(y_1)$ to avoid trivial attacks.
– Adversary $\mathcal{A}$ eventually returns a guess β' for a bit β and wins if $\beta' = \beta$.

A weaker version has been defined, when the messages y_0, y_1 for the challenge ciphertext are chosen before the Setup algorithm started, then the $\mathcal{T}-\mathcal{FE}$ scheme is said selectively-security against chosen-plaintext attacks, which is denoted by $\mathsf{sel}-\mathsf{IND}-\mathsf{CPA}$.

Traceability. The security game between the attacker $\mathcal{A}$ and the challenger $\mathcal{B}$ takes place as follows:

1. The challenger $\mathcal{B}$ runs $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and the public key PK sent to the adversary $\mathcal{A}$. $\mathcal{B}$ also creates a table $\mathcal{T}$ to store pairs of identities of users who queried keys and functions F, for all $F \in \mathcal{F}_\lambda$. It means that the table $\mathcal{T}$ stores (ID, F). Initially $\mathcal{T}$ is set empty.
2. The adversary adaptively makes secret key queries to the challenger. Concretely, the adversary $\mathcal{A}$ chooses some pairs of identities ID and functions $F \in \mathcal{F}_\lambda$ to query functional secret keys. The challenger $\mathcal{B}$ stores all these pairs in the table $\mathcal{T}$ and replies with the secret keys $\mathsf{sk}_{F,\mathsf{ID}}$ for those pairs.
3. The adversary $\mathcal{A}$ outputs $(F^*, \mathcal{D}_{F^*})$ and two messages y^0, y^1, where $\mathcal{D}_{F^*}$ is a pirate distinguisher for the function F^*.
4. After receiving the messages y^0, y^1 from $\mathcal{A}$, the challenger $\mathcal{B}$ runs the algorithm $\mathsf{Tracing}^{\mathcal{D}_{F^*}}(\mathsf{MSK}, F^*, 1^\mu, y^0, y^1)$ and outputs an identity ID^*.

We say that the adversary $\mathcal{A}$ wins the game if the output of $\mathsf{Tracing}$ is either an invalid symbol $\mathsf{ID}^* = \perp$ or the identity ID^* did not ask for F^*: $(\mathsf{ID}^*, F^*) \notin \mathcal{T}$.

When the adversary $\mathcal{A}$ is allowed to ask secret keys for the only target function F^* (but for any ID), and so for (ID, F^*), the security of Tracing algorithm will then be called *one-target security.*

As explained in the introduction, this one-target security also covers the case where the adversary outputs any function F such that the target function F^* is computable from F with public information. In such a case, when the pirate outputs the function F and the decoder $\mathcal{D}_F$ (together with two messages), one can define a decoder $\mathcal{D}_{F^*}$ that calls $\mathcal{D}_F$ and then applies the computation of F^* from F on the output, then do tracing on this $\mathcal{D}_{F^*}$, applying also the public transformation to the messages.

4 Our Inner-Product Functional Encryption

We will describe concretely a traceable functional encryption for inner product scheme $(\mathcal{T} - \mathcal{FE})$ for n users. Let $\mathbb{G}$ be a bilinear group of large prime of order q. Additionally, let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ denote a bilinear map, where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of order q, written multiplicatively.

$\mathsf{Setup}(1^\lambda, 1^k)$**:** This algorithm generates a bilinear setting $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e)$ for sufficiently large prime order q and g_1, g_2 respectively are generators of the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. The bilinear map e over $\mathbb{G}_1$, $\mathbb{G}_2$ can be calculated efficiently.

- Randomly choose $t_1, \ldots, t_k \xleftarrow{\$} \mathbb{Z}_q$, set $\boldsymbol{t} = (t_1, \ldots, t_k)$ and $b_1 = g_1^{t_1}, \ldots, b_k = g_1^{t_k}$.
- For each $i \in \{1, \ldots, k\}$, randomly choose $s_i \xleftarrow{\$} \mathbb{Z}_q$. We set $\boldsymbol{s} = (s_1, \ldots, s_k)$ and set $G = e(g_1, g_2) \in \mathbb{G}_T$ and $H_i = G^{s_i} \in \mathbb{G}_T$ for all $i = 1, \ldots, k$.
- We consider a linear code $\boldsymbol{\Gamma}$ over the alphabet $\mathbb{Z}_q$ with n codewords $\boldsymbol{\Gamma} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n\}$, corresponding to n users in our system. Each codeword has the length k.
- The public key is $\mathsf{PK} = \Big(\mathbb{G}, \boldsymbol{\Gamma}, g_1, g_2, G, H_1, \ldots, H_k, b_1, \ldots, b_k\Big)$.
- The master secret key is $\mathsf{MSK} = \{\boldsymbol{s}, \boldsymbol{t}\}$.

$\mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, \boldsymbol{x})$**:** Takes as input an identity ID, the master secret key MSK and a characteristic vector $\boldsymbol{x} = (x_1, \ldots, x_k) \in \mathbb{Z}_q^k$. Choose a (new) vector (codeword) $\boldsymbol{\theta}_{\mathsf{ID}} = (\theta_1, \ldots, \theta_k) \in \boldsymbol{\Gamma}$. A secret key is an element $g_2^{\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}}} \in \mathbb{G}_2$ such that $\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}} \cdot \boldsymbol{\theta}_{\mathsf{ID}}$ is a representation of $g_1^{\langle \boldsymbol{s}, \boldsymbol{x} \rangle}$ in the basis of $(b_1, b_2, \ldots, b_k)$. That is $g_1^{\langle \boldsymbol{s}, \boldsymbol{x} \rangle} = \prod_{i=1}^k b_i^{\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}} \theta_i} = b_1^{\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}} \theta_1} \cdots b_k^{\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}} \theta_k}$. Concretely, set $\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}} = \dfrac{\langle \boldsymbol{s}, \boldsymbol{x} \rangle}{\langle \boldsymbol{t}, \boldsymbol{\theta}_{\mathsf{ID}} \rangle}$ and define $\mathsf{sk}_{\boldsymbol{x},\mathsf{ID}} = g_2^{\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}}}$ for $\boldsymbol{\theta}_{\mathsf{ID}}$.

$\mathsf{Encrypt}(\mathsf{PK}, \boldsymbol{y})$**:** Takes as input the public key PK and a message $\boldsymbol{y} = (y_1, \ldots, y_k) \in \mathbb{Z}_q^k$. To encrypt $\boldsymbol{y}$, sample $r \xleftarrow{\$} \mathbb{Z}_q$ and compute

$$\mathsf{CT} = (H_1^r G^{y_1}, \ldots, H_k^r G^{y_k}, b_1^r, \ldots, b_k^r).$$

$\mathsf{Decrypt}(\mathsf{PK}, \mathsf{sk}_{\boldsymbol{x},\mathsf{ID}}, \mathsf{CT})$**:** Takes as input the public key PK, the secret key $\mathsf{sk}_{\boldsymbol{x},\mathsf{ID}} = g_2^{\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}}}$ for $\boldsymbol{\theta}_{\mathsf{ID}} = (\theta_1, \ldots, \theta_k)$ and a ciphertext CT, the algorithm computes

$$E = \frac{\left(H_1^r G^{y_1}\right)^{x_1} \cdots \left(H_k^r G^{y_k}\right)^{x_k}}{e\Big(\left(b_1^r\right)^{\theta_1} \cdots \left(b_k^r\right)^{\theta_k}, g_2^{\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}}}\Big)}.$$

Finally, it returns the discrete logarithm of E in basis $G = e(g_1, g_2)$.

Correctness: For all $(\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k)$, all $\boldsymbol{y} \in \mathbb{Z}_q^k$ and $\boldsymbol{x} \in \mathbb{Z}_p^k$, for $\mathsf{sk}_{\boldsymbol{x},\mathsf{ID}} = (g_2^{\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}}}, \boldsymbol{\theta}_{\mathsf{ID}}) \leftarrow \mathsf{Extract}(\mathsf{ID}, \mathsf{MSK}, \boldsymbol{x})$ and $\mathsf{CT} \leftarrow \mathsf{Encrypt}(\mathsf{PK}, \boldsymbol{y})$, we have that

$$\begin{aligned}\frac{\left(H_1^r G^{y_1}\right)^{x_1} \cdots \left(H_k^r G^{y_k}\right)^{x_k}}{e\Big(\left(b_1^r\right)^{\theta_1} \cdots \left(b_k^r\right)^{\theta_k}, g_2^{\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}}}\Big)} &= \frac{G^{\langle \boldsymbol{x},\boldsymbol{y}\rangle} \cdot \left(G^{x_1 s_1 + \cdots + x_k s_k}\right)^r}{e\Big(g_1^{t_1 r \theta_1} \cdots g_1^{t_k r \theta_k}, g_2^{\frac{\langle \boldsymbol{s},\boldsymbol{x}\rangle}{\langle \boldsymbol{t},\boldsymbol{\theta}_{\mathsf{ID}}\rangle}}\Big)} \\ &= \frac{G^{\langle \boldsymbol{x},\boldsymbol{y}\rangle} \cdot \cancel{G^{r\langle \boldsymbol{s},\boldsymbol{x}\rangle}}}{\cancel{e(g_1, g_2)^{r\langle \boldsymbol{s},\boldsymbol{x}\rangle}}} G^{\langle \boldsymbol{x},\boldsymbol{y}\rangle} = e(g_1, g_2)^{\langle \boldsymbol{x},\boldsymbol{y}\rangle}.\end{aligned}$$

Theorem 7. *The above $\mathcal{T} - \mathcal{FE}$ achieves the selective security (*$\mathsf{sel}-\mathsf{IND}-\mathsf{CPA}$*) under the BDDH assumption*

Proof. We assume that there exists an adversary $\mathcal{A}$ can distinguish distributions of ciphertexts in the real game with non-negligible advantage. We build a simulator $\mathcal{B}$ that solves the BDDH problem. It means that $\mathcal{B}$ takes as input a tuple $\left(g_1^a, g_1^b, g_2^a, g_2^c, T\right) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2 \times \mathbb{G}_T$, it must decide whether the input is BDDH tuple where $T = e\left(g_1, g_2\right)^{abc}$ or random tuple where $T = e\left(g_1, g_2\right)^z$. We set

$$\begin{aligned}\mathcal{D}_0 &= \left\{\left(g_1^a, g_1^b, g_2^a, g_2^c, e\left(g_1, g_2\right)^{abc}\right) | a, b, c \xleftarrow{\$} \mathbb{Z}_q\right\} \\ \mathcal{D}_1 &= \left\{\left(g_1^a, g_1^b, g_2^a, g_2^c, e\left(g_1, g_2\right)^z\right) | a, b, c, z \xleftarrow{\$} \mathbb{Z}_q\right\}.\end{aligned}$$

The algorithm $\mathcal{B}$ progresses as follows:

- Firstly, $\mathcal{B}$ is provided two distinct messages $\boldsymbol{y}_0$ and $\boldsymbol{y}_1$.
- $\mathcal{B}$ chooses $\boldsymbol{\Gamma} = \{\boldsymbol{\theta_1}, \ldots, \boldsymbol{\theta_n}\}$ is a linear code of size n and length k, as well as $t_1, \ldots, t_k \xleftarrow{\$} \mathbb{Z}_q$. Set $\boldsymbol{t} = (t_1, \ldots, t_k)$ and $b_i = g_1^{t_i}$, for $i = 1$ to k.
- $\mathcal{B}$ finds a $(k-1)$-basis of subspace $(\boldsymbol{y}_0 - \boldsymbol{y}_1)^\perp$ because the adversary $\mathcal{A}$ can only ask secret keys for vectors $\boldsymbol{x}$ in $(\boldsymbol{y}_0 - \boldsymbol{y}_1)^\perp$. We denote this basis by $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{k-1})$. For $i = 1, \ldots, k-1$, $\mathcal{B}$ randomly chooses $u_i \xleftarrow{\$} \mathbb{Z}_q$.
- We consider the canonical basis $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k)$ of $\mathbb{Z}_q^k$. A linear transformation from basis $(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{k-1}, (\boldsymbol{y}_0 - \boldsymbol{y}_1))$ to $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k)$ is given by: $\boldsymbol{e}_i = \alpha_i (\boldsymbol{y}_0 - \boldsymbol{y}_1) + \sum_{j=1}^{k-1} \lambda_{i,j} \boldsymbol{z}_j$, where the coefficients $\alpha_i, \lambda_{i,j}$ can be found efficiently by $\mathcal{B}$. Note that $\langle \boldsymbol{e}_i, \boldsymbol{y}_0 - \boldsymbol{y}_1\rangle = \alpha_i \times ||\boldsymbol{y}_0 - \boldsymbol{y}_1||^2$. Then $\boldsymbol{\alpha} = \sum_i \alpha_i \boldsymbol{e}_i = 1/||\boldsymbol{y}_0 - \boldsymbol{y}_1||^2 \times \sum_i \langle \boldsymbol{e}_i, \boldsymbol{y}_0 - \boldsymbol{y}_1\rangle \boldsymbol{e}_i = 1/||\boldsymbol{y}_0 - \boldsymbol{y}_1||^2 \times (\boldsymbol{y}_0 - \boldsymbol{y}_1)$.
- From the challenge tuple, and random scalars $u_1, \ldots, u_{k-1} \xleftarrow{\$} \mathbb{Z}_q$, set $G = e(g_1, g_2^c)$ and

$$\begin{aligned}H_i &= e\Big(\left(g_1^a\right)^{\alpha_i} \cdot g_1^{\sum_{j=1}^{k-1} u_j \lambda_{i,j}}, g_2^c\Big) \\ &= e(g_1, g_2^c)^{a\alpha_i + \sum_{j=1}^{k-1} u_j \lambda_{i,j}} = G^{a\alpha_i + \sum_{j=1}^{k-1} u_j \lambda_{i,j}}\end{aligned}$$

for $i = 1, \ldots, k$, which implicitly defines $s_i = a\alpha_i + \sum_{j=1}^{k-1} u_j \lambda_{i,j}$. The public key is set to $\mathsf{PK} = \Big(\mathbb{G}, \mathbf{\Gamma}, g_1, g_2, H_1, \ldots, H_k, b_1, \ldots, b_k\Big)$.

- For any vector $\boldsymbol{x} = (x_1, \ldots, x_k) \in (\boldsymbol{y}_0 - \boldsymbol{y}_1)^{\perp}$, $\mathcal{B}$ computes $\kappa_{\boldsymbol{x}} = \langle \boldsymbol{s}, \boldsymbol{x} \rangle = \sum_{j=1}^{k-1} \sum_{i=1}^{k} x_i u_j \lambda_{i,j}$ and, for identities ID, $\mathsf{sk}_{\boldsymbol{x},\mathsf{ID}} = g_2^{\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}}}$, where $\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}} = \dfrac{\kappa_{\boldsymbol{x}}}{\langle \boldsymbol{t}, \boldsymbol{\theta}_{\mathsf{ID}} \rangle}$. It sends the value $g_2^{\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}}}$ to $\mathcal{A}$. Vector θ_{ID} is a codeword in $\mathbf{\Gamma}$.
- The challenger randomly picks $\beta \xleftarrow{\$} \{0,1\}$ and, from the challenge tuple where T is the last element in $\mathbb{G}_T$, gives $\mathcal{A}$ a ciphertext $\mathsf{CT} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_{2k})$, where $\mathsf{ct}_j = T^{\alpha_j} \cdot e\Big(\big(g_1^b\big)^{\sum_{i=1}^{k-i} u_i \lambda_{j,i}}, g_2^c \Big) \cdot G^{y_{\beta,j}}$ and $\mathsf{ct}_{j+k} = \big(g_1^b\big)^{t_j}$, for $j = 1, \ldots, k$.
- At the end, the adversary outputs his guess β' for β. If $\beta' = \beta$ then $\mathcal{B}$ returns 1 for "BDDH tuple". Otherwise returns 0 for "random tuple". We will show that $\mathcal{B}$ can break BDDH assumption. To do so, we need to prove that the difference below is negligible

$$\begin{aligned} &\Big|\Pr[\mathcal{B}(\mathcal{D}_0) = 1] - \Pr[\mathcal{B}(\mathcal{D}_1) = 1]\Big| \\ = &\Big|\Pr[\beta = \beta' \mid T = e\,(g_1, g_2)^{abc}] - \Pr[\beta = \beta' \mid T = e\,(g_1, g_2)^{z}]\Big|. \end{aligned}$$

We find that:

1. When $T = e\,(g_1, g_2)^{abc}$ then we have $\mathsf{ct}_j = T^{\alpha_j} \cdot e\Big(\big(g_1^b\big)^{\sum_{i=1}^{k-1} u_i \lambda_{j,i}}, g_2^c \Big) \cdot G^{y_{\beta,j}} = H_j^b G^{y_{\beta,j}}$, for $j = 1, \ldots, k$. Therefore

$$\mathsf{CT} = \Big(H_1^b G^{y_{\beta,1}}, \ldots, H_k^b G^{y_{\beta,k}}, b_1^b, \ldots, b_k^b\Big).$$

   It implies that $\mathcal{B}$ perfectly simulates the real game. Since $\mathcal{A}$ can break the semantic security with non-negligible probability, we have $\Pr[\beta = \beta' \mid T = e\,(g_1, g_2)^{abc}] = \mathsf{Adv}(\mathcal{A}) + 1/2$.
2. When $T = e(g_1, g_2)^z = G^v$ is random element, the challenger will send $\mathcal{A}$ the ciphertext of message $\boldsymbol{y}_\beta + v\boldsymbol{\alpha} = \boldsymbol{y}_\beta + v/||\boldsymbol{y}_0 - \boldsymbol{y}_1||^2 \times (\boldsymbol{y}_0 - \boldsymbol{y}_1) = \mu \boldsymbol{y}_0 + (1-\mu)\boldsymbol{y}_1$, for some random $\mu \in \mathbb{Z}_q$. This makes β perfectly unpredictable: $\Pr[\beta = \beta' \mid T = e\,(g_1, g_2)^{z}] = 1/2$.

We conclude the advantage is non-negligible as

$$\begin{aligned} &\Big|\Pr[\beta = \beta' \mid T = e\,(g_1, g_2)^{abc}] - \Pr[\beta = \beta' \mid T = e\,(g_1, g_2)^{z}]\Big| \\ = &\Big|\mathsf{Adv}(\mathcal{A}) + \frac{1}{2} - \frac{1}{2}\Big| = \mathsf{Adv}(\mathcal{A}). \end{aligned}$$

5 Black-Box Confirmation Traitor-Tracing

This section will be devoted to present a black-box confirmation traitor-tracing algorithm. The purpose of this algorithm is to verify sets of secret keys which are suspected by a Tracer. The tracing algorithm takes as input the master secret key MSK and it can access the table $\mathcal{T}$ (see the Tracing security game) to take a set of secret keys for which it wants to check its suspicion. We will use the scalar form $\mathsf{tk}_{\boldsymbol{x},\mathsf{ID}}$ of the secret keys instead of the group element form $\mathsf{sk}_{\boldsymbol{x},\mathsf{ID}}$. But as we only consider possible legitimate secrete keys in this form, the scalars are known to the authority.

5.1 Notations

Suppose that Tracer is provided a set of t secret keys (for the suspected traitors), say $\mathcal{K}_{\mathsf{suspect}} = \{\mathsf{tk}_1, \ldots, \mathsf{tk}_t\}$ which are derived from a fixed vector $\boldsymbol{x} = (x_1, \ldots, x_k)$. Here, we have slightly

abused the notation, as we are in the one-target security. When the vector $\boldsymbol{x}$ is explicit, we use the notation $\{\mathsf{tk}_1, \ldots, \mathsf{tk}_t\}$ instead of $\{\mathsf{tk}_{\boldsymbol{x},1}, \ldots, \mathsf{tk}_{\boldsymbol{x},t}\}$, the pirate decoder $\mathcal{D}_{\boldsymbol{x}}$ is replaced by $\mathcal{D}$ and we use integers to represent identities of users. A codeword will be $\boldsymbol{\theta}_i$ which is attached to a user with identity i. The goal of the Tracer is to verify whether there is any traitor in $\mathcal{K}_{\mathsf{suspect}}$. Before go further we need to define some notations.

- Set $\mathcal{K}_i = \{\mathsf{tk}_1, \ldots, \mathsf{tk}_i\} \subseteq \mathcal{K}_{\mathsf{suspect}}$, for all $i \in [t]$ and $\mathcal{K}_0 = \emptyset$.
- We define spaces of tracing signals (ciphertexts) $\mathsf{Tr}_0, \mathsf{Tr}_1, \ldots, \mathsf{Tr}_t$ such that each signal from Tr_i can be decrypted successfully by any secret key in $\mathcal{K}_i$. More concretely, for each i from 0 to t, the tracing signal for a message $\boldsymbol{y} = (y_1, \ldots, y_k)$ is taken from the distribution $\mathsf{Tr}_i^{\boldsymbol{x}}(\boldsymbol{y})$ (or $\mathsf{Tr}_i(\boldsymbol{y})$ for simplicity, when $\boldsymbol{x}$ is explicit) that is defined as follows

$$\left\{ \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, g_1^{z_1}, \ldots, g_1^{z_k} \right) \;\middle|\; \begin{array}{l} a \xleftarrow{\$} \mathbb{Z}_q, \boldsymbol{z} \xleftarrow{\$} \mathbb{Z}_q^k, \\ \langle \boldsymbol{z}, \mathsf{tk}_j \boldsymbol{\theta}_j \rangle = a \langle \boldsymbol{s}, \boldsymbol{x} \rangle, \forall j \in [i] \end{array} \right\},$$

  where $\boldsymbol{z} = (z_1, \ldots, z_k)$. $G, H_1, \ldots, H_k$ are group elements of $\mathbb{G}_T$ and belong to the public key PK. Set $\mathcal{Q}(a) = e(g_1, g_2)^{a\langle \boldsymbol{s}, \boldsymbol{x} \rangle}$, as $\boldsymbol{s}$ and $\boldsymbol{x}$ are fixed.
- Every user j with secret key in $\mathcal{K}_i$ can output the same

$$\frac{(H_1^a G^{y_1})^{x_1} \cdots (H_k^a G^{y_k})^{x_k}}{e(g_1^{\langle \boldsymbol{z}, \boldsymbol{\theta}_j \rangle}, g_2^{\mathsf{tk}_j})} = \frac{\mathcal{P}(\boldsymbol{y}, a)}{\mathcal{Q}(a)},$$

  where $\mathcal{P}(\boldsymbol{y}, a) = (H_1^a G^{y_1})^{x_1} \cdots (H_k^a G^{y_k})^{x_k} = \mathcal{Q}(a) \times G^{\langle \boldsymbol{y}, \boldsymbol{x} \rangle}$.
- Define distribution of normal ciphertext for a message $\boldsymbol{y} = (y_1, \ldots, y_k)$, denoted $\mathsf{Norm}(\boldsymbol{y})$: randomly draw $r \xleftarrow{\$} \mathbb{Z}_q$ and output ciphertext $(H_1^r G^{y_1}, \ldots, H_k^r G^{y_k}, b_1^r, \ldots, b_k^r)$.
- For $i = 0, \ldots, t$, we set $p_i = \Pr[\mathcal{D}(\mathsf{CT}) = b \,|\, b \xleftarrow{\$} \{0,1\}, \mathsf{CT} \leftarrow \mathsf{Tr}_i(\boldsymbol{y}_b)]$, where $\boldsymbol{y}_0, \boldsymbol{y}_1$ are chosen by $\mathcal{D}$. When $i = 0$, in $\mathsf{Tr}_i(\boldsymbol{y_b})$, a and $\boldsymbol{z}$ are perfectly independent, and so under the DDH assumption, the H_i^a hides the $y_{b,i}$. So we have $p_0 = 1/2 + \mathsf{negl}(\lambda)$.

Definition 8. A tracing traitor algorithm is black-box confirmation if it satisfies:

1. Confirmation: If suspected set of users actually contains the entire set of traitors then output of Tracing algorithm always returns at least an identity i such that $\mathsf{tk}_i \in \mathcal{K}_{\mathsf{suspect}}$ is guilty. Formally, with the condition $\mathcal{K}_{\mathcal{D}} \subseteq \mathcal{K}_{\mathsf{suspect}}$, the Tracing algorithm returns at least an identity i such that the secret key $\mathsf{tk}_i \in \mathcal{K}_{\mathsf{suspect}}$ as guilty. We denote by $\mathcal{K}_{\mathcal{D}}$ a set of secret keys used to build the pirate decoder $\mathcal{D}$.
2. Soundness: The honest users will never be accused if the Tracing algorithm outputs an identity as guilty; it is impossible for traitors to deceive Tracing algorithm to blame innocent users. Said differently if Tracing algorithm outputs an identity i such that tk_i is guilty then $\mathsf{tk}_i \in \mathcal{K}_{\mathcal{D}}$.

5.2 Tracing Algorithm

Tracing algorithm needs to use following lemmas.

Lemma 9. *Under the DDH assumption in $\mathbb{G}_1$, no adversary corruping t users $1, \ldots, t$ can distinguish the distribution of tracing signals $\mathsf{Tr}_t(\boldsymbol{y})$ with the distribution of normal ciphertexts $\mathsf{Norm}(\boldsymbol{y})$, for any adversarially chosen $\boldsymbol{y}$.*

Proof. Suppose that an adversary $\mathcal{A}$ can distinguish the distribution of tracing signals $\mathsf{Tr}_t(\boldsymbol{y})$ with the distribution of normal ciphertexts Norm. We will build a simulator $\mathcal{B}$ breaks the DDH assumption in $\mathbb{G}_1$. The simulator has inputs: 4-tuples $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \in \mathbb{G}_1^4$, where $\mathfrak{g}_2 = \mathfrak{g}_1^c$ and c is unknown. It decides whether this is a DDH tuple or a random tuple:

1. Take randomly t codewords $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t$ from the code $\boldsymbol{\Gamma}$.
2. Take randomly A from $\mathbb{Z}_q$ such that $\mathfrak{g}_1^A \mathfrak{g}_2 \neq 1$.
3. Take randomly $\boldsymbol{a} = (a_1, \ldots, a_k), \boldsymbol{e} = (e_1, \ldots, e_k) \xleftarrow{\$} \mathbb{Z}_q^k$ such that $\langle \boldsymbol{\theta}_i, \boldsymbol{a} - A\boldsymbol{e} \rangle = 0$, for all $i = 1, \ldots, t$.
4. Set $b_i = \mathfrak{g}_1^{a_i} \mathfrak{g}_2^{e_i}$, for all $i = 1, \ldots, k$.
5. Take randomly $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k) \xleftarrow{\$} \mathbb{Z}_q^k$ such that $\langle \boldsymbol{\alpha}, \boldsymbol{a} - A\boldsymbol{e} \rangle = 0$. Take randomly $g_2 \xleftarrow{\$} \mathbb{G}_2$ and it sets $g_1 = \mathfrak{g}_1^A \mathfrak{g}_2$, $G = e(g_1, g_2)$. We set $H_i = e(u_1^A u_2, g_2)^{\alpha_i}$ for all $i \in [k]$. The public key is $\mathsf{PK} = (\mathbb{G}, \boldsymbol{\Gamma}, g_1, g_2, G, H_1, \ldots, H_k, b_1, \ldots, b_k)$, where $\mathbb{G}$ is a bilinear group.
6. The simulator $\mathcal{B}$ calculates secret key for queries $(\boldsymbol{x}, i)$, $\mathsf{tk}_{\boldsymbol{x},i} = \dfrac{\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle}{\langle \boldsymbol{\theta_i}, \boldsymbol{e} \rangle}$, for $i \in [t]$ and functions $\boldsymbol{x}$ then gives all $g_2^{\mathsf{tk}_{\boldsymbol{x},i}}$ to the adversary $\mathcal{A}$. It is clear that $\mathsf{tk}_{\boldsymbol{x},i} \boldsymbol{\theta_i}$ is a representation of $(\mathfrak{g}_1^A \mathfrak{g}_2)^{\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle}$ in the base $(b_1, \ldots, b_k)$.
7. Take randomly $a \xleftarrow{\$} \mathbb{Z}_q$. The simulator constructs the ciphertext for a message $\boldsymbol{y}$ as below

$$\mathsf{CT} = (H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, (u_1^{a_1} u_2^{e_1})^a, \ldots, (u_1^{a_k} u_2^{e_k})^a),$$

   where $\boldsymbol{y} = (y_1, \ldots, y_k)$.
8. Send the ciphertext CT to the adversary $\mathcal{A}$. If $\mathcal{A}$ decides the ciphertext comes from normal distribution (i.e. $\mathcal{A}$ returns 1) then $\mathcal{B}$ returns "DDH tuple", else returns "random tuple".

We first show that the public key PK which is generated by the simulator $\mathcal{B}$ is indistinguishable from the corresponding public key in the real algorithm.

– We will prove that distribution of tuples $(b_1, \ldots, b_k) \in \mathbb{G}_1^k$ is uniform. Indeed, , write $b_i = g_1^{t_i}$ then, for each $(t+k)$-tuple $(\mathbf{0}, t_1, \ldots, t_k)$ where $t_1, \ldots, t_k \xleftarrow{\$} \mathbb{Z}_q$ and $\mathbf{0} = (0, \ldots, 0) \in \mathbb{Z}_q^t$ the below system of equations has a solution

$$\left(\begin{array}{ccc|ccc} \ldots & \boldsymbol{\theta}_1 & \ldots & \ldots & -A\boldsymbol{\theta}_1 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \ldots & \boldsymbol{\theta}_t & \ldots & \ldots & -A\boldsymbol{\theta}_t & \ldots \\ \hline 1 & \ldots & 0 & c & \ldots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 1 & 0 & \ldots & c \end{array}\right) \times \left(\frac{\boldsymbol{a}}{\boldsymbol{e}}\right) = \left(\frac{\mathbf{0}}{\begin{array}{c} t_1 \\ \vdots \\ t_k \end{array}}\right).$$

  We denote by $\boldsymbol{\Gamma}_0$ a matrix with its rows are vectors $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t$. The rank of this matrix is t. Indeed, it is equivalent that

$$\begin{pmatrix} \boldsymbol{\Gamma}_0 & -A\boldsymbol{\Gamma}_0 \\ \mathbf{I}_k & c\mathbf{I}_k \end{pmatrix} \times \left(\frac{\boldsymbol{a}}{\boldsymbol{e}}\right) = \left(\frac{\mathbf{0}}{\boldsymbol{t}}\right)$$

  has solutions. Here $\mathbf{I}_k$ is the $(k \times k)$-unit matrix. We set

$$\boldsymbol{\Omega} = \begin{pmatrix} \boldsymbol{\Gamma}_0 & -A\boldsymbol{\Gamma}_0 \\ \mathbf{I}_k & c\mathbf{I}_k \end{pmatrix}.$$

  Since A is chosen such that $1 \neq \mathfrak{g}_1^A \mathfrak{g}_2 = \mathfrak{g}_1^{A+c}$, $((t+k) \times 2k)$-matrix $\boldsymbol{\Omega}$ has rank $k+t$. Therefore, dim $\mathsf{Im}\boldsymbol{\Omega}$ = rank $\boldsymbol{\Omega}$ and the dimension of $\mathsf{Ker}\boldsymbol{\Omega} = 2k - (k+t) = k - t$. Therefore, the above system of linear equations with unkowns $(\boldsymbol{a}, \boldsymbol{e})$ exists a solution. It implies that $(b_1, \ldots, b_k)$ is uniform over $\mathbb{G}_1^k$.
– Concerning H_i, in the real game $H_i = e(g_1, g_2)^{\alpha_i}$ for randomly chosen but known g_1, g_2 while in the simulation game, $H_i = e(u_1^A u_2, g_2)^{\alpha_i}$ for randomly chosen A and $(\alpha_i)_i$ in a span of dimension $k-1$. Under the DDH in the $\mathbb{G}_1$, $u_1^A u_2$ is indistinguishable from random, and thus H_i follows from a correct distribution in the computational sense.

We now show that, for any adversarially chosen $\boldsymbol{y}$, if $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \in \mathbb{G}_1^4$ is a DDH tuplee then the ciphertext is a normal ciphertext of $\boldsymbol{y}$ and when it is a random tuple then the ciphertext comes from $\mathsf{Tr}_t(\boldsymbol{y})$. Therefore, if the adversary can distinguish these two distributions then $\mathcal{B}$ can break the DDH assumption in $\mathbb{G}_1$: $|\Pr[\mathcal{B}(\mathcal{D}_0) = 1] - \Pr[\mathcal{B}(\mathcal{D}_1) = 1]|$ is non-negligible. By definition, it is equivalent to

$$\left|\Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid (\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \xleftarrow{\$} \mathcal{D}_0] - \Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid (\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \xleftarrow{\$} \mathcal{D}_1]\right|$$

is non-negligible. Here, CT is a ciphertext generated as in Step 7. We find that:

1. When $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \xleftarrow{\$} \mathcal{D}_0$, we will prove that

$$\Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid (\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \xleftarrow{\$} \mathcal{D}_0] = \Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid \mathsf{CT} \xleftarrow{\$} \mathsf{Norm}].$$

Indeed, suppose that $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) = (\mathfrak{g}_1, \mathfrak{g}_2, \mathfrak{g}_1^z, \mathfrak{g}_2^z)$, where z is unknown. The ciphertexts in Step 7 is then:

$$\begin{aligned}
\mathsf{CT} &= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, (u_1^{a_1} u_2^{e_1})^a, \ldots, (u_1^{a_k} u_2^{e_k})^a\right) \\
&= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, (\mathfrak{g}_1^{za_1} \mathfrak{g}_2^{ze_1})^a, \ldots, (\mathfrak{g}_1^{za_k} \mathfrak{g}_2^{ze_k})^a\right) \\
&= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, (\mathfrak{g}_1^{a_1} \mathfrak{g}_2^{e_1})^{z \cdot a}, \ldots, (\mathfrak{g}_1^{a_k} \mathfrak{g}_2^{e_k})^{z \cdot a}\right) \\
&= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, b_1^{z \cdot a}, \ldots, b_k^{z \cdot a}\right),
\end{aligned}$$

which is in the space of normal ciphertext. It is sufficient thus to show that, with the decryption with the secret key $\mathsf{tk}_{\boldsymbol{x},i}$, the decryption will gives $G^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}$. Indeed,

$$\begin{aligned}
E &= \frac{\left(H_1^a G^{y_1}\right)^{x_1} \cdots \left(H_k^a G^{y_k}\right)^{x_k}}{e\left(\left(u_1^{a_1} u_2^{e_1}\right)^{a\theta_1} \cdots \left(u_1^{a_k} u_2^{e_k}\right)^{a\theta_k}, g_2^{\mathsf{tk}_{\boldsymbol{x},i}}\right)} \\
&= \frac{G^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot e(u_1^A u_2, g_2)^{ax_1\alpha_1} \cdots e(u_1^A u_2, g_2)^{ax_k\alpha_k}}{e\left(\left(\mathfrak{g}_1^A \mathfrak{g}_2\right)^{za\langle \boldsymbol{e}, \boldsymbol{\theta} \rangle}, g_2^{\frac{\langle \boldsymbol{x}, \boldsymbol{\alpha} \rangle}{\langle \boldsymbol{\theta}, \boldsymbol{e} \rangle}}\right)} \\
&= \frac{G^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot e(\mathfrak{g}_1^A \mathfrak{g}_2, g_2)^{azx_1\alpha_1} \cdots e(\mathfrak{g}_1^A \mathfrak{g}_2, g_2)^{azx_k\alpha_k}}{e\left(\left(\mathfrak{g}_1^A \mathfrak{g}_2\right)^{za\langle \boldsymbol{e}, \boldsymbol{\theta} \rangle}, g_2^{\frac{\langle \boldsymbol{x}, \boldsymbol{\alpha} \rangle}{\langle \boldsymbol{\theta}, \boldsymbol{e} \rangle}}\right)} \\
&= \frac{G^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot e(\mathfrak{g}_1^A \mathfrak{g}_2, g_2)^{az\langle \boldsymbol{x}, \alpha \rangle}}{e\left(\left(\mathfrak{g}_1^A \mathfrak{g}_2\right)^{za\langle \boldsymbol{e}, \boldsymbol{\theta} \rangle}, g_2^{\frac{\langle \boldsymbol{x}, \boldsymbol{\alpha} \rangle}{\langle \boldsymbol{\theta}, \boldsymbol{e} \rangle}}\right)} = G^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}.
\end{aligned}$$

2. When $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \xleftarrow{\$} \mathcal{D}_1$, we will prove that

$$\Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid (\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \xleftarrow{\$} \mathcal{D}_1] = \Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid \mathsf{CT} \xleftarrow{\$} \mathsf{Tr}_t].$$

Indeed, suppose that $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) = (\mathfrak{g}_1, \mathfrak{g}_2, \mathfrak{g}_1^{\gamma_1}, \mathfrak{g}_2^{\gamma_2})$, where $\gamma_1 \neq \gamma_2$ and $\mathfrak{g}_2 = \mathfrak{g}_1^c$. The ciphertexts in Step 7 is then:

$$\begin{aligned}
\mathsf{CT} &= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, (u_1^{a_1} u_2^{e_1})^a, \ldots, (u_1^{a_k} u_2^{e_k})^a\right) \\
&= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, (\mathfrak{g}_1^{\gamma_1 a_1} \mathfrak{g}_2^{\gamma_2 e_1})^a, \ldots, (\mathfrak{g}_1^{\gamma_1 a_k} \mathfrak{g}_2^{\gamma_2 e_k})^a\right) \\
&= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, \mathfrak{g}_1^{a(\gamma_1 a_1 + c\gamma_2 e_1)}, \ldots, \mathfrak{g}_1^{a(\gamma_1 a_k + c\gamma_2 e_k)}\right) \\
&= \left(H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, \mathfrak{g}_1^{z_1}, \ldots, \mathfrak{g}_1^{z_k}\right),
\end{aligned}$$

where $z_i = a(\gamma_1 a_k + c\gamma_2 e_k)$ for all $i \in [k]$.
We show that for any traitor with the key $\mathsf{tk}_{\boldsymbol{x},i}$, $i = 1$ to t, it decrypts to the same message. Indeed:

$$\begin{aligned}
E &= \frac{\left(H_1^a G^{y_1}\right)^{x_1} \cdots \left(H_k^a G^{y_k}\right)^{x_k}}{e\Big(\left(\mathfrak{g}_1^{\gamma_1 a_1}\mathfrak{g}_2^{\gamma_2 e_1}\right)^{a\theta_1} \cdots \left(\mathfrak{g}_1^{\gamma_1 a_k}\mathfrak{g}_2^{\gamma_2 e_k}\right)^{a\theta_k}, g_2^{\mathsf{tk}_{\boldsymbol{x},i}}\Big)} \\
&= \frac{G^{\langle \boldsymbol{x},\boldsymbol{y}\rangle} \cdot e(u_1^A u_2, g_2)^{a x_1 \alpha_1} \cdots e(u_1^A u_2, g_2)^{a x_k \alpha_k}}{e\Big(\mathfrak{g}_1^{\langle \boldsymbol{a},\boldsymbol{\theta}\rangle a\gamma_1}\mathfrak{g}_2^{\langle \boldsymbol{e},\boldsymbol{\theta}\rangle a\gamma_2}, g_2^{\frac{\langle \boldsymbol{x},\boldsymbol{\alpha}\rangle}{\langle \boldsymbol{\theta},\boldsymbol{e}\rangle}}\Big)} \\
&= \frac{G^{\langle \boldsymbol{x},\boldsymbol{y}\rangle} \cdot e(\mathfrak{g}_1^{A\gamma_1}\mathfrak{g}_2^{\gamma_2}, g_2)^{a x_1 \alpha_1} \cdots e(\mathfrak{g}_1^{A\gamma_1}\mathfrak{g}_2^{\gamma_2}, g_2)^{a x_k \alpha_k}}{e\Big(\mathfrak{g}_1^{\langle \boldsymbol{e},\boldsymbol{\theta}\rangle aA\gamma_1}\mathfrak{g}_2^{\langle \boldsymbol{e},\boldsymbol{\theta}\rangle a\gamma_2}, g_2^{\frac{\langle \boldsymbol{x},\boldsymbol{\alpha}\rangle}{\langle \boldsymbol{\theta},\boldsymbol{e}\rangle}}\Big)} \\
&= \frac{G^{\langle \boldsymbol{x},\boldsymbol{y}\rangle} \cdot e(\mathfrak{g}_1^{A\gamma_1}\mathfrak{g}_2^{\gamma_2}, g_2)^{a\langle \boldsymbol{x},\alpha\rangle}}{e\Big(\left(\mathfrak{g}_1^{A\gamma_1}\mathfrak{g}_2^{\gamma_2}\right)^{a\langle \boldsymbol{e},\boldsymbol{\theta}\rangle}, g_2^{\frac{\langle \boldsymbol{x},\boldsymbol{\alpha}\rangle}{\langle \boldsymbol{\theta},\boldsymbol{e}\rangle}}\Big)} = \frac{G^{\langle \boldsymbol{x},\boldsymbol{y}\rangle} \cdot e(\mathfrak{g}_1^{A\gamma_1}\mathfrak{g}_2^{\gamma_2}, g_2)^{a\langle \boldsymbol{x},\alpha\rangle}}{e\left(\mathfrak{g}_1^{A\gamma_1}\mathfrak{g}_2^{\gamma_2}, g_2\right)^{a\langle \boldsymbol{x},\boldsymbol{\alpha}\rangle}} = G^{\langle \boldsymbol{x},\boldsymbol{y}\rangle}.
\end{aligned}$$

Here $\boldsymbol{\theta}_i = (\theta_1, \ldots, \theta_k)$.
Finally, we will prove that the distribution of ciphertext CT is uniform over the space of signals Tr_t. It requires that the system of equations

$$\begin{pmatrix} \boldsymbol{\Gamma}_0 & -A\boldsymbol{\Gamma}_0 \\ \mathbf{I}_k & c\mathbf{I}_k \\ a\gamma_1\mathbf{I}_k & ac\gamma_2\mathbf{I}_k \end{pmatrix} \times \begin{pmatrix} \boldsymbol{a} \\ \boldsymbol{e} \end{pmatrix} = \boldsymbol{\gamma}$$

is consistent, where $\boldsymbol{\gamma}$ is a fixed vector in $\mathbb{Z}_q^{t+2k}$. It is equivalent that the below $(t+2k, 2k)$-matrix

$$\left(\begin{array}{ccc|ccc}
\ldots & \boldsymbol{\theta}_1 & \ldots & \ldots & -A\boldsymbol{\theta}_1 & \ldots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\ldots & \boldsymbol{\theta}_t & \ldots & \ldots & -A\boldsymbol{\theta}_t & \ldots \\
\hline
1 & \ldots & 0 & c & \ldots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \ldots & 1 & 0 & \ldots & c \\
\hline
a\gamma_1 & \ldots & 0 & ac\gamma_2 & \ldots & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
0 & \ldots & a\gamma_1 & 0 & \ldots & ac\gamma_2
\end{array}\right)$$

has full rank (i.e. rank $= 2k$). Indeed, This is straightforward because the last $2k$ rows of the above matrix are linear independent due to $\gamma_1 \neq \gamma_2$.

We conclude that

$$\begin{aligned}
&\big|\Pr\ [\ \mathcal{B}(\mathcal{D}_0) = 1] - \Pr[\mathcal{B}(\mathcal{D}_1) = 1]\ \big| \\
&\quad = |\Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid \mathsf{CT} \xleftarrow{\$} \mathsf{Normal}] - \Pr[\mathcal{A}(\mathsf{CT}) = 1 \mid \mathsf{CT} \xleftarrow{\$} \mathsf{Tr}_t]| = \mathsf{Adv}(\mathcal{A}),
\end{aligned}$$

which is non-negligible. □

Lemma 10 (Hybrid Lemma). *Considering the one-target security for an adversarially chosen target function $\boldsymbol{x}$. Under the DDH assumption over group $\mathbb{G}_1$, for all $1 \leq i_0 \leq t$, no adversary can distinguish the distribution of tracing signals $\mathsf{Tr}_{i_0}(\boldsymbol{y})$ with the distribution of $\mathsf{Tr}_{i_0-1}(\boldsymbol{y})$ unless it owns the secret key sk_{i_0}.*

Proof. Suppose that an adversary $\mathcal{A}$ can distinguish the distribution of tracing signals Tr_{i_0} with Tr_{i_0-1}. We build a simulator $\mathcal{B}$ that breaks DDH assumption. The simulator has input a 4-tuple $(\mathfrak{g}_1, \mathfrak{g}_2, u_1, u_2) \in \mathbb{G}_1^4$, where $\mathfrak{g}_2 = \mathfrak{g}_1^c$ and c is unknown. It must output "DDH tuple" or "random tuple".

1. Take randomly t codewords $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_t$ from the code $\boldsymbol{\Gamma}$ corresponding to t traitors and also take t codewords $\boldsymbol{\theta}_1^{(s)}, \ldots, \boldsymbol{\theta}_t^{(s)}$ from the code $\boldsymbol{\Gamma}$ corresponding to t suspected users. We are considering the adversary $\mathcal{A}$ does not know the secret key sk_{i_0} or the pirate decoder does not contain $\mathsf{sk}_{\boldsymbol{x},i_0}$ in itself.
2. Take randomly A from $\mathbb{Z}_q$ such that $\mathfrak{g}_1^A \mathfrak{g}_2 \neq 1$.
3. Take randomly $\boldsymbol{a} = (a_1, \ldots, a_k), \boldsymbol{e} = (e_1, \ldots, e_k) \xleftarrow{\$} \mathbb{Z}_q^k$ such that $\langle \boldsymbol{\theta}_i, \boldsymbol{a} - A\boldsymbol{e} \rangle = 0$, for all $i = 1, \ldots, t$, $\langle \boldsymbol{\theta}_i^{(s)}, \boldsymbol{a} - A\boldsymbol{e} \rangle = 0$, for all $i = 1, \ldots, t$, $i \neq i_0$ and $\langle \boldsymbol{\theta}_{i_0}^{(s)}, \boldsymbol{a} - A\boldsymbol{e} \rangle \neq 0$.
4. Set $b_i = \mathfrak{g}_1^{a_i} \mathfrak{g}_2^{e_i}$, for all $i = 1, \ldots, k$. Take randomly $\boldsymbol{v} = (v_1, \ldots, v_k) \xleftarrow{\$} \mathbb{Z}_q^k$ such that $\langle \boldsymbol{\theta}_i^{(s)}, \boldsymbol{v} \rangle = 0$, for all $i = 1, \ldots, i_0$.
5. Take randomly $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k) \xleftarrow{\$} \mathbb{Z}_q^k$ such that $\langle \boldsymbol{\alpha}, \boldsymbol{a} - A\boldsymbol{e} \rangle = 0$ and $\langle \boldsymbol{\alpha}, \boldsymbol{v} \rangle = 0$.
6. When $\mathcal{B}$ receives a target function $\boldsymbol{x}$ from $\mathcal{A}$. It calculates $\tau_i = \dfrac{\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle}{\langle \boldsymbol{\theta_i}, \boldsymbol{e} \rangle}$, for $i = 1, \ldots, t$ and then give all $g_2^{\tau_i}$ to the adversary $\mathcal{A}$ to create a Pirate Decoder $\mathcal{D}_{\boldsymbol{x}}$. Moreover, $\tau_i^{(s)} = \dfrac{\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle}{\langle \boldsymbol{\theta}_i^{(s)}, \boldsymbol{e} \rangle}$, for $i = 1, \ldots, i_0 - 1$. It is clear that $\tau_i \boldsymbol{\theta_i}$ and $\tau_i^{(s)} \boldsymbol{\theta}_i^{(s)}$ are representations of $(\mathfrak{g}_1^A \mathfrak{g}_2)^{\langle \boldsymbol{\alpha}, \boldsymbol{x} \rangle}$ in the base $(b_1, \ldots, b_k)$.
7. Take randomly $G, H_1, \ldots, H_k \xleftarrow{\$} \mathbb{G}_T$, $a \xleftarrow{\$} \mathbb{Z}_q$. When the simulator receives a message $\boldsymbol{y}$, it constructs the ciphertext

$$\mathsf{CT} = (H_1^a G^{y_1}, \ldots, H_k^a G^{y_k}, \mathfrak{g}_1^{v_1} u_1^{a_1} u_2^{e_1}, \ldots, \mathfrak{g}_1^{v_k} u_1^{a_k} u_2^{e_k}),$$

where $\boldsymbol{y} = (y_1, \ldots, y_k)$. It then sends the ciphertext to the adversary $\mathcal{A}$. If $\mathcal{A}$ returns the ciphertext comes from $\mathsf{Tr}_{i_0}(\boldsymbol{y})$ distribution then $\mathcal{B}$ returns DDH tuple, else returns random tuple.

By the similar argument as in Lemma 9, the ciphertext CT in Step 7 of the algorithm $\mathcal{B}$ comes from the distribution $\mathsf{Tr}_{i_0}(\boldsymbol{y})$ if the input of $\mathcal{B}$ is actually DDH tuples and from the distribution $\mathsf{Tr}_{i_0-1}(\boldsymbol{y})$ otherwise. □

Based on the lemmas 9 and 10, we can design a tracing algorithm that relies on the linear technique tracing:

- Initial step: Tracer constructs distributions of tracing signal $\mathsf{Tr}_t, \ldots, \mathsf{Tr}_0$.
- Do experiments on the pirate distinguisher $\mathcal{D}$ finitely many times. We start testing $\mathcal{D}$ by taking tracing signals CT from the distribution Tr_t. We measure the rate that $\mathcal{D}$ outputs correctly his guess, denoted by $\tilde{p}_t$. Experiments can be done because we can prove that the pirate distinguisher cannot distinguish distributions Tr_t and Norm (see Lemma 9).
- At step i, for $i = t - 1, \ldots, 0$. We do experiment on the pirate distinguisher $\mathcal{D}$ with tracing signals taken from Tr_i. From Lemma 10, the pirate distinguisher cannot see any change from previous step $i + 1$ to this step i unless it holds the secret key tk_{i+1}. More formally, we also measure the rate $\tilde{p}_i$ that $\mathcal{D}$ outputs correctly his guess and show that if $\mathcal{D}$ does not contain tk_{i+1} then there is no significant difference between $\tilde{p}_{i+1}$ and $\tilde{p}_i$.
- At the final step, $\mathcal{D}$ will be tested with tracing signals taken from Tr_0. $\mathcal{D}$ answers correctly only negligibly close to $1/2$.
- We output the traitor i such that the gap between $\tilde{p}_i$ and $\tilde{p}_{i-1}$ is the largest value among all indices i.

Below, we present the tracing algorithm in more details. We note that $\boldsymbol{y}_0, \boldsymbol{y}_1$ are vectors which are chosen by the pirate distinguisher $\mathcal{D}$.

For $i = t$ downto 0, do the following:

1. Let $\mathsf{cnt} \leftarrow 0$.
2. For $j = 1$ to $N = 8\lambda t^2/\mu$, do the following:
   i. $b \xleftarrow{\$} \{0,1\}$.
   ii. $\mathsf{CT} \xleftarrow{\$} \mathsf{Tr}_i(\boldsymbol{y}_b)$.
   iii. Send CT to $\mathcal{D}$. If $\mathcal{D}(\mathsf{CT}) = b$ then $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$.
3. End for.
4. Let $\tilde{p}_i$ be the fraction of times that $\mathcal{D}$ did the correct guess. We have $\tilde{p}_i = \mathsf{cnt}/N$.

End for.

Output identities i such that $|\tilde{p}_i - \tilde{p}_{i-1}| \geq \dfrac{\mu(\lambda)}{4t}$.

Below, we state and prove confirmation and soundness property of our Tracing algorithm.

Lemma 11 (Confirmation property). *The* Tracing *algorithm has the confirmation property under the DDH assumption in* $\mathbb{G}_1$.

Proof. We want to prove that in the case of that all the traitors are in the set of suspected users, i.e. $\mathcal{K}_{\mathcal{D}} \subseteq \mathcal{K}_{\mathsf{suspect}}$, the Tracing algorithm always returns the identity of a guilty. It means that the output of Tracing algorithm is not empty with high probability. We denote $\mathcal{A}$ an adversary who used the secret keys in $\mathcal{K}_{\mathcal{D}}$ to output the pirate distinguisher $\mathcal{D}$. Since the adversary $\mathcal{A}$ can create a μ-useful pirate distinguisher $\mathcal{D}$, it implies that $\left|p_{\mathsf{Norm}} - \frac{1}{2}\right| \geq \mu(\lambda)$, where

$$p_{\mathsf{Norm}} = \mathsf{Pr}\left[\begin{array}{c} (\mathsf{MSK}, \mathsf{PK}) \leftarrow \mathsf{Setup}(\cdot) \\ \{\mathsf{sk}_i \leftarrow \mathsf{Extract}(i, \mathsf{MSK}, \boldsymbol{x})\}_{i\in[n]} \\ \mathcal{D}\left(\mathsf{CT}_b\right) = b : (\mathcal{D}, \boldsymbol{y}^0, \boldsymbol{y}^1) \leftarrow \mathcal{A}(\mathsf{PK}, \{\mathsf{sk}_i\}_{i\in[t]}) \\ \text{st. } \langle \boldsymbol{x}, \boldsymbol{y}^0\rangle \neq \langle \boldsymbol{x}, \boldsymbol{y}^1\rangle \\ b \xleftarrow{\$} \{0,1\}, \mathsf{CT}_b \leftarrow \mathsf{Norm}(\mathsf{PK}, \boldsymbol{y}^b) \end{array}\right].$$

We denote S the set of indices $i \in [t]$ such that $|p_i - p_{i-1}| > \mu(\lambda)/4t$. The set S is well defined in the sense that $S \neq \emptyset$. Indeed, as we know that p_0 is negligibly close to $1/2$, and Lemma 9 showed that no adversary $\mathcal{A}$ can distinguish the distribution Norm from Tr_t, then $|p_t - p_0| \geq \mu(\lambda) - \mathsf{negl}(\lambda) > \mu(\lambda)/2$. Then, there exists an index i such that $|p_i - p_{i-1}| > \mu(\lambda)/2t$. Thus S is a non empty set. Applying Chernoff bound for all $i \in S$, we have on experimental probabilities

$$\Pr\left[|\tilde{p}_i - \tilde{p}_{i-1}| < \frac{\mu(\lambda)}{4t}\right] \leq \mathsf{negl}(\lambda),$$

Therefore, with overwhelming probability, there exists an index i such that

$$|\tilde{p}_i - \tilde{p}_{i-1}| \geq \frac{\mu(\lambda)}{4t}.$$

The latter is thus returned with overwhelming probability.

□

Lemma 12 (Soundness property). *The* Tracing *algorithm has the soundness property under the DDH assumption in* $\mathbb{G}_1$.

Proof. We now prove the soundness property of Tracing algorithm. Suppose that the Tracing algorithm outputs an identity j, where $\mathsf{tk}_j \in \mathcal{K}_{\mathsf{suspect}}$, we will prove that $\mathsf{tk}_j \in \mathcal{K}_{\mathcal{D}}$. According to Chernoff bound, thanks to $N = 8\lambda t^2/\mu(\lambda)$ to calculate $\tilde{p}_i$, for all i, we have

$$\Pr\left[|\tilde{p}_i - p_i| > \frac{\mu(\lambda)}{16t}\right] < 2 \cdot e^{-\lambda/64}.$$

Therefore, with high probability we have $|\tilde{p}_i - p_i| \leq \mu(\lambda)/16t$, for all $i = 0, \ldots, t$.

By definition, whenever the Tracing algorithm outputs j as a guilty, we have $|\tilde{p}_j - \tilde{p}_{j-1}| \geq \mu(\lambda)/4t$, and thus $|p_j - p_{j-1}| \geq \mu(\lambda)/8t$. In other words, the pirate distinguisher can distinguish the two tracing signals Tr_j and Tr_{j-1} with advantage at least $\mu(\lambda)/8t$. It implies that $\mathcal{D}$ contains the secret key tk_j, $\mathsf{tk}_j \in \mathcal{D}$. This follows from the fact that if $\mathcal{D}$ does not know the secret key tk_j, $\mathsf{tk}_j \notin \mathcal{D}$, the two tracing signals Tr_j and Tr_{j-1} are indistinguishable. More concretely, under the hardness of the DDH problem in group $\mathbb{G}_1$, it is impossible for the pirate to distinguish Tr_j and Tr_{j-1} without tk_j. This is stated and proved in Lemma 10. □

Theorem 13. *Under the DDH assumption, our tracing scheme is one-target security in black-box confirmation model.*

Proof. We recall that in the black-box confirmation model we will verify a set suspected secret keys $\mathcal{K}_{\mathsf{suspect}} = \{\mathsf{tk}_1, \ldots, \mathsf{tk}_t\}$ which are also derived from the vector $\boldsymbol{x}$. We will prove that Tracing algorithm always outputs an identity of a traitor whenever $\mathcal{K}_{\mathcal{D}} \cap \mathcal{K}_{\mathsf{suspect}} \neq \emptyset$. It means that Tracer always wins in the game with the pirate distinguisher $\mathcal{D}$. Indeed, we consider the following two cases:

- In the first case $\mathcal{K}_{\mathcal{D}} \subseteq \mathcal{K}_{\mathsf{suspect}}$. It means that all traitors are in suspicious set $\mathcal{K}_{\mathsf{suspect}}$. Tracing algorithm will output a guilty identity i by the confirmation property. According to soundness property, the identity is a traitor ($\mathsf{tk}_i \in \mathcal{K}_{\mathcal{D}}$).
- In case $\mathcal{K}_{\mathcal{D}} \not\subseteq \mathcal{K}_{\mathsf{suspect}}$ and $\mathcal{K}_{\mathcal{D}} \cap \mathcal{K}_{\mathsf{suspect}} \neq \emptyset$. Because $\mathcal{K}_{\mathcal{D}} \cap \mathcal{K}_{\mathsf{suspect}} \neq \emptyset$, tracing algorithm will output an identity i so that $\mathsf{sk}_i \in \mathcal{K}_{\mathsf{suspect}}$. It implies i is a traitor ($\mathsf{tk}_i \in \mathcal{K}_{\mathcal{D}}$) by the soundness property. □

Acknowledgments

This work was supported in part by the European Community's Seventh Framework Programme (FP7/2007-2013 Grant Agreement no. 339563 – CryptoCloud) and the ANR ALAMBIC (ANR16-CE39-0006).

References

ABDP15. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Heidelberg, March / April 2015. (Pages 1, 2, 3, and 4.)

ABP+17. Shweta Agrawal, Sanjay Bhattacherjee, Duong Hieu Phan, Damien Stehlé, and Shota Yamada. Efficient public trace and revoke from standard assumptions: Extended abstract. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 17*, pages 2277–2293. ACM Press, October / November 2017. (Page 2.)

ADML+07. Michel Abdalla, Alexander W. Dent, John Malone-Lee, Gregory Neven, Duong Hieu Phan, and Nigel P. Smart. Identity-based traitor tracing. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 361–376. Springer, Heidelberg, April 2007. (Page 2.)

AHL+08. Man Ho Au, Qiong Huang, Joseph K. Liu, Willy Susilo, Duncan S. Wong, and Guomin Yang. Traceable and retrievable identity-based encryption. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *ACNS 08*, volume 5037 of *LNCS*, pages 94–110. Springer, Heidelberg, June 2008. (Page 2.)

ALS16. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016. (Page 1.)

BCFG17. Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 67–98. Springer, Heidelberg, August 2017. (Page 1.)

BF99. Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 338–353. Springer, Heidelberg, August 1999. (Pages 1, 2, and 3.)

BSW06. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592. Springer, Heidelberg, May / June 2006. (Page 2.)

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011. (Page 1.)

BZ14. Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Heidelberg, August 2014. (Page 2.)

CFN94. Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270. Springer, Heidelberg, August 1994. (Page 1.)

CPP05. Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 542–558. Springer, Heidelberg, May 2005. (Page 2.)

CVW+18. Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from LWE made simple and attribute-based. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 341–369. Springer, Heidelberg, November 2018. (Page 2.)

FNP07. Nelly Fazio, Antonio Nicolosi, and Duong Hieu Phan. Traitor tracing with optimal transmission rate. In Juan A. Garay, Arjen K. Lenstra, Masahiro Mambo, and René Peralta, editors, *ISC 2007*, volume 4779 of *LNCS*, pages 71–88. Springer, Heidelberg, October 2007. (Page 2.)

GKW18. Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 660–670. ACM Press, June 2018. (Pages 2, 4, and 6.)

Goy07. Vipul Goyal. Reducing trust in the PKG in identity based cryptosystems. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 430–447. Springer, Heidelberg, August 2007. (Page 2.)

KY02. Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 450–465. Springer, Heidelberg, April / May 2002. (Page 2.)

LCW13. Zhen Liu, Zhenfu Cao, and Duncan S. Wong. Blackbox traceable CP-ABE: How to catch people leaking their keys by selling decryption devices on eBay. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13*, pages 475–486. ACM Press, November 2013. (Page 2.)

LPSS14. San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k-LWE and applications in traitor tracing. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 315–334. Springer, Heidelberg, August 2014. (Page 2.)

LW15a. Zhen Liu and Duncan S. Wong. Practical ciphertext-policy attribute-based encryption: Traitor tracing, revocation, and large universe. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 15*, volume 9092 of *LNCS*, pages 127–146. Springer, Heidelberg, June 2015. (Page 2.)

LW15b. Zhen Liu and Duncan S. Wong. Traceable CP-ABE on prime order groups: Fully secure and fully collusion-resistant blackbox traceable. In Sihan Qing, Eiji Okamoto, Kwangjo Kim, and Dongmei Liu, editors, *ICICS 15*, volume 9543 of *LNCS*, pages 109–124. Springer, Heidelberg, December 2015. (Page 2.)

NCD+14. Jianting Ning, Zhenfu Cao, Xiaolei Dong, Lifei Wei, and Xiaodong Lin. Large universe ciphertext-policy attribute-based encryption with white-box traceability. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS 2014, Part II*, volume 8713 of *LNCS*, pages 55–72. Springer, Heidelberg, September 2014. (Page 2.)

NWZ16. Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 388–419. Springer, Heidelberg, May 2016. (Page 2.)

PT11. Duong Hieu Phan and Viet Cuong Trinh. Identity-based trace and revoke schemes. In Xavier Boyen and Xiaofeng Chen, editors, *ProvSec 2011*, volume 6980 of *LNCS*, pages 204–221. Springer, Heidelberg, October 2011. (Page 2.)

SW05. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005. (Page 1.)