Privacy-Preserving and Traceable Functional Encryption for Inner Product In Cloud Computing

Muyao Qiu, Jinguang Han, *Senior Member, IEEE*, Feng Hao, *Senior Member, IEEE*, Chao Sun, and Ge Wu

***Abstract*—Cloud computing is a distributed infrastructure that centralizes server resources on a platform in order to provide services over the internet. Traditional public-key encryption protects data confidentiality in cloud computing, while functional encryption provides a more fine-grained decryption method, which only reveals a function of the encrypted data. However, functional encryption in cloud computing faces the problem of key sharing. In order to trace malicious users who share keys with others, traceable FE-IP (TFE-IP) schemes were proposed where the key generation center (KGC) knows users' identities and binds them with different secret keys. Nevertheless, existing schemes fail to protect the privacy of users' identities. The fundamental challenge to construct a privacy-preserving TFE-IP scheme is that KGC needs to bind a key with a user's identity without knowing the identity. To balance privacy and accountability in cloud computing, we propose the concept of privacy-preserving traceable functional encryption for inner product (PPTFE-IP) and give a concrete construction which offers the features: (1) To prevent key sharing, both a user's identity and a vector are bound together in the key; (2) The KGC and a user execute a two-party secure computing protocol to generate a key without the former knowing anything about the latter's identity; (3) Each user can ensure the integrity and correctness of his/her key through verification; (4) The inner product of the two vectors embedded in a ciphertext and in his/her key can be calculated by an authorized user; (5) Only the tracer can trace the identity embedded in a key. We formally reduce the security of the proposed PPTFE-IP to well-known complexity assumptions, and conduct an implementation to evaluate its efficiency. The novelty of our scheme is to protect the user's privacy and provide traceability if required.**

***Index Terms*—Functional Encryption, Inner Production, Traceability, Privacy, Security.**

I. Introduction

CLOUD computing allows users to outsource data to remote servers and access data from anywhere at anytime. When users outsource sensitive data to cloud servers, encryption is a reliable solution to protect data confidentiality in the cloud [1], [2], [3]. Traditional public-key encryption only allows for an all-or-nothing decryption approach, meaning that the decrypted data can either reveal all of the original information or nothing at all. Functional encryption (FE) introduces a novel paradigm of public-key encryption, in which decryption only reveals the function value of encrypted data [4].

An instance of functional encryption is functional encryption for inner product (FE-IP), which only computes the inner product of two vectors related to a user's key and a ciphertext without revealing anything else. FE-IP has numerous practical applications, such as cloud computing [5], [6], machine learning [7], [8], [9], federated learning [10], [11], [12] and Internet of Things [13], [14], etc.

In existing FE-IP schemes [15], [16], a central authority (CA) is required to generate secret keys for users. Hence, the CA is assumed be fully honest; otherwise, it can impersonate any user to decrypt ciphertexts and may release users' personal information. Han et al. [17] presented a privacy-preserving FE-IP scheme in which the CA and a user execute a two-party secure computing protocol to generate a key without the former knowing anything about the latter's identity. Although this approach provides full anonymity in the key issuing process, it does not address the traceability problem. Therefore, if a user shares his/her secret key with others, he/she cannot be identified. To monitor malicious key leakage while protecting user privacy, we propose the first privacy-preserving traceable functional encryption for inner product (PPTFE-IP) scheme. In PPTFE-IP, a tracer is introduced to identify the identity of a key holder when he/she shares his/her key with others. However, the CA knows nothing about a user's identity.

A. Related Work

In this section, we review related schemes.

1) Functional Encryption: Functional encryption (FE) [4] is a novel paradigm in public-key encryption, in which decryption only reveals a function of the encrypted data without revealing anything else. The concept of FE was firstly proposed by Boneh et al. [4], who formalized the definition and security model of this encryption technique.

Abdalla et al. [15] introduced the concept of functional encryption with inner product (FE-IP), where the decryption process only outputs the inner product of two vectors associated to a ciphertext and a user's secret key, and presented a simple FE-IP scheme with selective security. Building on this work, Abdalla et al. [18] constructed a novel generic scheme, which is secure against adaptive adversaries. Agrawal et al. [16] introduced fully secure FE-IP schemes based on the same assumptions as [15] and gave a simple method to achieve the bounded collusion FE for all circuits.

In above FE-IP schemes, CA must be trusted fully because it generates all the secret keys of users. This is known as

This work was supported by the National Natural Science Foundation of China (Grant No. 62372103, 61972190), the Natural Science Foundation of Jiangsu Province (Grant No. BK20231149), the Jiangsu Provincial Scientific Research Center of Applied Mathematics (Grant No. BK20233002), the Fundamental Research Funds for the Central Universities (Grant No. 2242021R40011) and the Start-up Research Fund of Southeast University(Grant No. RF1028623300). *(Corresponding author: Jinguang Han.)*

M. Yao, J. Han, C. Sun and G. Wu are with the School of Cyber Science and Engineering, Southeast University, Nanjing 210096, China, and J. Han is also with the Engineering Research Center of Blockchain Application, Supervision and Management (Southeast University), Minister of Education, Nanjing 210096, China (e-mail: {myqiu, jghan,sunchaomt,gewu}@seu.edu.cn).

H. Feng is with Department of Computer Science, University of Warwick, Coventry CV4 7AL, United Kingdom (e-mail: feng.hao@warwick.ac.uk).

the key escrow problem [19]. Some FE-IP schemes were proposed mainly to reduce trust on the CA and solve the key escrow problem. Abdalla et al. [20] presented a decentralized FE-IP to transform any scheme with key-derivation into a decentralized key-derivation scheme. A new primitive called decentralized multi-client FE-IP was proposed by Chotard et al. [21] where clients generate ciphertexts non-interactively, and secret keys are generated in a decentralized way. Jérémy et al. [22] proposed the first dynamic decentralized FE-IP, which allows participants to join dynamically during the lifetime of a system. In these schemes, multiple authorities or clients, instead of a single entity, issue keys to users. To reduce trust on the CA and protect privacy, Han et al. [17] introduced a decentralized privacy-preserving FE-IP scheme, in which a user and multiple authorities collaborate to generate a secret key, and the authorities do not know the identity associated with the key.

2) Traitor Tracing: Traitor tracing schemes are able to trace the identity of a key holder who leaked the secret key [23]. There are two types of traitor tracing: white-box tracing and black-box tracing. In white box tracing [24], [25], [26], a malicious user is traced given a well-formed secret key. In black box tracing [27], [28], the malicious users are traced using a black box (including an unknown key and algorithm) which can decrypt ciphertexts. Some schemes [27] introduced black-box traceable ciphertext-policy attribute-based encryption (CP-ABE) to trace the malicious user; however, these schemes were impractical because a composite-order group is required. In order to overcome the problem, Xu et al. [28] presented a CP-ABE scheme based on the prime-order group supporting black-box traceability.

A user's identity is embedded in his/her secret key in white-box tracing schemes [24], [25]. In [24], the tracer generates part of keys for users and records some auxiliary information for trace. Han et al. [25] used a similar method as [24] but a binary tree was applied to reduce the cost of storing users' identities and corresponding information. In [26], if the secret key is leaked or abused, any entity can trace the identity embedded in the key, this is called public traceability. However, restricting tracing to only the tracer is essential to ensure user privacy. To curb key leakage issues and trace users in FE-IP, [29] defined a novel primitive called traceable functional encryption for inner product (TFE-IP) and proposed a concrete black-box tracing scheme for FE-IP. Following [29], Luo et al. [30] introduced the first efficient traceable FE-IP scheme supporting public, black-box tracing and revocation, which achieves adaptive security (A-IND-CPA) under standard assumptions. Luo et al. [30] also proposed the first generic TFE-IP scheme that achieves adaptive security. Dutta et al. [31] introduced fully collusion resistant TFE-IP schemes for the first time which are public and black-box traceable, and gave generic constructions of TFE-IP based on both pairing and lattices. Branco et al. [32] introduced a new traceable FE-IP model based on registered FE where users generate their secret-public key pairs and register their public keys.

However, a traceable FE-IP scheme with private traceability has not been considered. Also, users' privacy and anonymity may be violated by traceability. To balance the relationship between privacy and traceability, we propose a PPTFE-IP scheme. Though the concept and purpose of tracing in attribute-based encryption are similar to traceable functional encryption, challenges encountered and techniques adopted vary widely. The comparison of different features between our scheme and the existing schemes is illustrated in Table I. Our scheme can recover a user's identity through a well-formed secret key, while other schemes can trace a user's identity via decryption boxes; therefore, our scheme is white-box traceable, while other schemes are black-box traceable. Among existing schemes, schemes [30], [31], [33], [32] are publicly traceable, which undermines user privacy. [29] provides similar trace functionality with our scheme, namely, only the tracer can trace the identity embedded in a key. Therefore, [29] and our scheme are private tracing; on the contrary, [30], [31], [33], [32] are public tracing. In [17], to protect users' privacy, the two-party secure computing technique was applied to embed identities in secret keys. Since the KGC knows nothing about users' identities, it is difficult to extract the identities of users from their secret keys. In [29], [30], [31], [32] and [33], to provide traceability, users' identities were bound with secret keys. If we directly apply the two-party secure computing technique to these schemes to construct a PPTFE-IP scheme, the traceability cannot be guaranteed. The novelty of our scheme is to protect users' privacy and provide traceability if required.

TABLE I
COMPARISON WITH EXISTING WORK

Scheme	Traceability	Public/Private Traceability	Privacy-Preservation
[29]	Black box	Private	–
[30]	Black box	Public	–
[31]	Black box	Public	–
[32]	Black box	Public	–
[33]	Black box	Public	–
Ours	**White box**	**Private**	✓

B. Our Contributions

Our PPTFE-IP scheme balances the relationship between traceability and anonymity in FE-IP. Specifically, the PPTFE-IP scheme offers some interesting features:

1) To prevent key sharing, both a user's identity and a vector are bound together in the key;
2) The key generation center (KGC) and a user execute a two-party secure computing protocol to generate a key with the KGC knowing nothing about the user's identity;
3) Each user can ensure the integrity and correctness of his/her key through verification;
4) The inner product of the two vectors embedded in a ciphertext and in his/her key can be calculated by an authorized user;
5) Only the tracer can trace the identity embedded in a key.

The contributions of this paper are shown below:

1) We formalize the definitions and security models of PPTFE-IP.
2) A concrete TFE-IP scheme is constructed on the symmetric bilinear groups.

3) A privacy-preserving key generation (PPKeyGen) algorithm is proposed for TFE-IP scheme.
4) An implementation is conducted to evaluate the efficiency of our PPTFE-IP.
5) We formally reduce the security of the proposed PPTFE-IP to well-known complexity assumptions.

C. Techniques and Challenges

It is nontrivial to construct a PPTFE-IP based on existing schemes. When constructing our PPTFE-IP scheme, we encounter the following challenges:

1) To prevent key sharing and realize tracing, it is necessary to embed a user's identity in his/her key. However, there is a risk that the user's identity is leaked to the KGC, threatening the privacy of the user.
2) We require a new traceable FE-IP scheme which supports privacy-preserving key generation.
3) It is challenging to prevent user collusion attacks.

To address these challenges, we employ the following technologies:

1) To protect user's identity, the two-party secure computing technique is applied to enable the KGC to derive a key to an anonymous user without knowing his/her identity. During key generation, the user's identity and the generated secret key are unknown to KGC.
2) To enable the tracer to trace the identity of a key holder, a user's identity is encrypted under the public key of the tracer by using a verifiable public-key encryption scheme. For privacy preservation, during key generation, the user sends the ciphertext to KGC and proves that he knows the identity included in the ciphertext. KGC verifies the identity through the zero-knowledge proof while not knowing the user's true identity.
3) To prevent the combination of secret keys, all elements in a key are bound together by a random number. Hence, two users cannot collude to combine their secret keys.

D. Organization

We offer an overview of the preliminaries as well as the formal definition and security models of PPTFE-IP in Section II. Section III presents the concrete construction of our PPTFE-IP scheme. Section IV offers a detailed security proof for PPTFE-IP. We conduct a comparative analysis of our scheme with existing TFE-IP schemes, implement and evaluate our scheme in Section V. At last, we conclude this paper and outline future work in Section VI.

II. Preliminaries

In this section, we present preliminaries along with formal definitions and security models utilized in this paper. The symbols employed in this paper are outlined in Table II.

Figure 1 depicts the framework of our PPTFE-IP scheme. The solid lines represent the necessary flow of the entire scheme, and the dash lines represents a special case that trace is required. id_* represents the identity of the user who shared his/her secret key with outers. Our scheme works as follows: At first, the KGC sets up the system, and generates the master secret key and public parameters. Then, the Data Owner encrypts his data by using the public parameters and uploads ciphertexts to the Cloud Server. To obtain a secret key form the KGC without releasing his identity id_i, the user U_i first executes a privacy-preserving secret key generation algorithm with the KGC. An authorized user can use his secret key to decrypt the ciphertext and obtain the inner product $\langle \vec{f}, \vec{m} \rangle$. In the case that traceability is required, the tracer can identity the real identity of the user who shared his secret key with others.

TABLE II
NOTATION SUMMARY

Notation	Description
DL	Discrete logarithm
DDH	Decisional Diffie-Hellman
q-SDH	q-strong Diffie-Hellman
1^λ	A security parameter
PP	Public parameters
FE	Functional encryption
FE-IP	FE for inner product
TFE-IP	Traceable FE-IP
PPTFE-IP	Privacy-preserving TFE-IP
PPKeyGen	Privacy-preserving key generation algorithm
KGC	Key generation center
$\vec{x}$	A vector
$\langle \vec{x}, \vec{y} \rangle$	The inner product of $\vec{x}$ and $\vec{y}$
$\epsilon(\lambda)$	A negligible function in λ
$\perp$	Empty
ζ	Failure
com	Commitment
$decom$	Decommitment
θ	User's identity
$\mathcal{A}$	A probabilistically polynomial time adversary
$\mathcal{GG}$	A group generation algorithm
$\mathcal{BG}$	A bilinear group generation algorithm
$\mathcal{C}$	A challenger
$\mathcal{S}$	A simulator

A. Bilinear Groups and Complexity Assumptions

Definition 1 (Prime Order Bilinear Groups): Let $\mathbb{G}$ be a (multiplicative) cyclic group generated by $g \in \mathbb{G}$ with prime order p and $\mathbb{G}_T$ be a (multiplicative) cyclic group of prime order p. If the following conditions are satisfied, a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear pairing:

1 Bilinearity. $\forall a, b \in \mathbb{Z}_p, g_1, g_2 \in \mathbb{G}, e(g_1^a, g_2^b) = e(g_1^b, g_2^a) = e(g_1, g_2)^{a \cdot b}$;
2 Non-Generation. Let 1 be the identity element in $\mathbb{G}_T$. $\forall g_1, g_2 \in \mathbb{G}$, $e(g_1, g_2) \neq 1$;
3 Computability. $\forall g_1, g_2 \in \mathbb{G}$, $e(g_1, g_2)$ can be calculated efficiently.

Definition 2 (Decisional Diffie-Hellman(DDH) Assumption[34]): $\mathcal{GG}(1^\lambda) \to (\mathbb{G}, p, g)$, where λ is security parameter, $\mathbb{G}$ is a group generated by $g \in \mathbb{G}$ with prime order p. Given α, β, γ randomly chosen from $\mathbb{Z}_p$, DDH assumption holds on $(p, \mathbb{G})$ if the tuples $\left(g^\alpha, g^\beta, g^{\alpha\beta}\right)$ and $\left(g^\alpha, g^\beta, g^\gamma\right)$ are computationally indistinguishable by all adversaries $\mathcal{A}$ with a negligible advantage $\epsilon(\lambda)$, in other words,

$$Adv_{\mathcal{A}}^{DDH} = \left| \Pr\left[\mathcal{A}(g^\alpha, g^\beta, g^{\alpha\beta}) = 1\right] - \Pr\left[\mathcal{A}(g^\alpha, g^\beta, g^\gamma) = 1\right] \right|$$

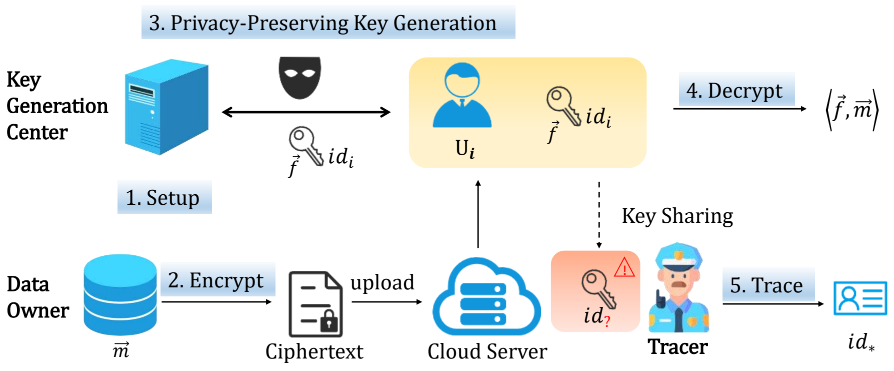


Fig. 1. The Framework of Our PPTFE-IP Scheme

$\leq \epsilon(\lambda)$.

Definition 3 (Discrete Logarithm (DL) Assumption [35]): $\mathcal{GG}(1^\lambda) \to (\mathbb{G}, p, g)$, $\mathbb{G}$ is a group generated by $g \in \mathbb{G}$ with prime order p. Given g, y, the DL assumption is satisfied on the group $(p, \mathbb{G})$, all $\mathcal{A}$ have the negligible advantage $\epsilon(\lambda)$ in computing $x \in \mathbb{Z}_p$ from $y = g^x$, namely

$$Adv_{\mathcal{A}}^{DL} = \left| \Pr\left[y = g^x | \mathcal{A}(g, y) \to x) \right] \right| \leq \epsilon(\lambda).$$

Definition 4 (q-Strong Diffie-Hellman(q-SDH) Assumption[36]): $\mathcal{GG}(1^\lambda) \to (e, p, \mathbb{G}, \mathbb{G}_T)$. Suppose that g is a generator of the group $\mathbb{G}$. q-SDH assumption holds on $(e, p, \mathbb{G}, \mathbb{G}_T)$ if given a (q+1)-tuple $\left(g, g^x, g^{(x^2)}, \cdots, g^{(x^q)}\right)$ as input, all adversaries $\mathcal{A}$ output a pair $\left(c, g^{\frac{1}{x+c}}\right)$ with the negligible advantage, namely

$$\begin{aligned} &Adv_{\mathcal{A}}^{q-SDH} \\ &= \left| \Pr\left[\mathcal{A}\left(g, g^x, g^{(x^2)}, \cdots, g^{(x^q)}\right) \to \left(c, g^{\frac{1}{x+c}}\right) \right] \right| \\ &\leq \epsilon(\lambda), \text{ where } x \xleftarrow{R} \mathbb{Z}_p, c \in \mathbb{Z}_p \text{ and } c \neq -x. \end{aligned}$$

B. Formal Definition

In a TFE-IP scheme, the tracer can trace the identity of the corresponding key owner from the secret key. We follow the definition introduced in [29] and [24] to define our TFE-IP. Firstly, the definition and security model of our TFE-IP scheme are formalized, then the formal definition and security model of our PPTFE-IP scheme are presented.

Definition 5 (TFE-IP): A TFE-IP scheme is formally defined by the five algorithms as follows:

1 **Setup**$(1^\lambda) \to (msk, Tk, PP)$. The Setup algorithm takes a security parameter 1^λ as input and outputs the master secret key msk, trace key Tk and public parameters PP.

2 **Encrypt**$(PP, \vec{m}) \to Ct$. The Encryption algorithm inputs public parameters and a vector $\vec{m}$, and produces the ciphertext Ct as output.

3 **KeyGen**$\left(msk, \vec{f}, id\right) \to sk_{\vec{f},id}$. The KeyGen algorithm takes master secret key msk, a vector $\vec{f}$ and a user's identity id as input, and outputs $sk_{\vec{f},id}$ as user's secret key for function $\vec{f}$.

4 **Decrypt**$\left(Ct, sk_{\vec{f},id}, id\right) \to \langle \vec{f}, \vec{m} \rangle$ *or* $\perp$. The Decryption algorithm takes ciphertext Ct, user's secret key $sk_{\vec{f},id}$ and the user's identity id as input, and outputs the inner product value. If the decryption failed, the algorithm outputs $\perp$.

5 **Trace**$\left(sk_{\vec{f},id}, Tk\right) \to id$ *or* $\perp$. The Tracing algorithm takes user's secret key $sk_{\vec{f},id}$ and the tracer's trace key as input, and outputs the user's identity id or failure $\perp$.

C. Security Model

1) s-IND-CPA Security: To define the security, we apply the selective indistinguishability against the chosen plaintext attacks (s-IND-CPA) model. The game below executed between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ is applied to define this s-IND-CPA model.

- **Initialization.** Two vectors $\vec{m_0}, \vec{m_1}$ with the same length are submitted by the adversary $\mathcal{A}$.
- **Setup.** The challenger $\mathcal{C}$ runs $(1^\lambda) \to (msk, PP)$, returns PP to $\mathcal{A}$ and creates a set V which is initially empty.
- **Phase-I (KeyGen Query) .** $\mathcal{A}$ submits an identity id and a vector $\vec{f}$ limiting $\langle \vec{f}, \vec{m_0} \rangle = \langle \vec{f}, \vec{m_1} \rangle$. $\mathcal{C}$ executes $KeyGen(msk, \vec{f}, id) \to sk_{\vec{f},id}$, and returns $sk_{\vec{f},id}$. $\mathcal{C}$ updates $V \leftarrow V \cap \{\vec{f}, id\}$. The adversary $\mathcal{A}$ queries multiple times.
- **Challenge.** $\mathcal{C}$ randomly selects a $\delta \in \{0, 1\}$. $\mathcal{C}$ executes $Enc(PP, \vec{m_\delta}) \to Ct^*$, and returns Ct^* to $\mathcal{A}$.
- **Phase-II. Phase-I** is repeated.
- **Output.** $\mathcal{A}$ guesses δ' about δ. In the case that $\delta' = \delta$, $\mathcal{A}$ wins the game.

Definition 6 (s-IND-CPA Security): A traceable functional encryption for inner product scheme is secure in the s-IND-CPA model if any adversary $\mathcal{A}$ has a negligible advantage $\epsilon(\lambda)$ in winning the above game, namely

$$Adv_{\mathcal{A}} = \left| \Pr[\delta' = \delta] - \tfrac{1}{2} \right| < \epsilon(\lambda).$$

2) Traceability: For traceability, we follow the security model proposed in [24], [37], [38]. This security model is described by the following game executed by a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

- **Setup.** $\mathcal{C}$ executes $\mathbf{Setup}\left(1^{\lambda}\right)$, and sends public parameters PP to $\mathcal{A}$.
- **Key Query.** $\mathcal{A}$ submits $(id, \vec{f})$ in which id represents the user identity and $\vec{f}$ represents the function. $\mathcal{C}$ returns the secret key to $\mathcal{A}$. $\mathcal{A}$ can query multiple times.
- **Trace Query.** $\mathcal{A}$ sends a key $(sk_i)_{i\in[r]} = (K_{1i}, K_{2i}, K_{3i}, K_{4i}, K_{5i})$ to $\mathcal{C}$. $\mathcal{C}$ runs $Trace\left(sk_{\vec{f},id}, tsk, PP\right) \rightarrow id$, and sends id to $\mathcal{A}$. $\mathcal{A}$ can query multiple times.
- **Key Forgery.** $\mathcal{A}$ forges and produces sk_* as output. Assume the identity embedded in sk_* is id_*. $\mathcal{A}$ wins the game if $Trace\left(sk_*, tsk, PP\right) \neq id_*$ or $Trace\left(sk_*, tsk, PP\right) \notin \{\perp, id_1, id_2, \cdots, id_q\}$.

Definition 7 (Traceability): A TFE-IP scheme is fully traceable if any adversary $\mathcal{A}$ wins the above game with a negligible advantage $\epsilon(\lambda)$, namely

$$\begin{aligned} &Adv_{\mathcal{A}} \\ &\quad = \Pr\Big[\{Trace\left(sk^*, tsk, PP\right) \notin \{\perp, id_1, id_2, \cdots, id_q\}\} \vee \\ &\quad \{Trace\left(sk^*, tsk, PP\right) \neq id_*\}\Big] < \epsilon(\lambda). \end{aligned}$$

3) PPTFE-IP: A PPTFE-IP scheme comprises the same **Setup, Encryption, Decryption, Trace** algorithms as the TFE-IP scheme mentioned in Section II-B. However, the **KeyGen** algorithm in the TFE-IP scheme is replaced by the **PPKeyGen** algorithm, which is described below.

PPKeyGen$(User(PP, id, \vec{f}, decom_u) \leftrightarrow KGC(PP, msk, \vec{f}, com_u) \rightarrow (\perp, sk_{\vec{f},id})$. This algorithm includes the interaction process between the user and the KGC. Let $Commitment(PP, id) \rightarrow (com_u, decom_u)$ be a commitment scheme which inputs the public parameters and a secret identity id, producing the commitment $comm_u$ and decommitment $decom_u$ as output. Given the public parameters PP, the master secret key msk, a vector $\vec{f}$ and the commitment com_u, KGC outputs $\perp$. The user takes the public parameters PP, an identity id, a vector $\vec{f}$ and $decom_u$ as input. If $decommitment(PP, id, decom_u, com_u) = 1$, a secret key $sk_{\vec{f},id}$ is output; Otherwise, the output fails. The formalization of the security model for PPKeyGen algorithm employs two games [39], [40]: $selective - failure - blind$ and $leakage - free$.

Selective-failure-blind. The user U is honest and the KGC is malicious in this game. KGC tries to distinguish the user's identity id associated with the key.

- KGC publishes public parameters PP and submits two identities id_0, id_1.
- KGC randomly selects $\delta \in \{0,1\}$ and gets two commitments $com_\delta, com_{1-\delta}$ which belongs to id_0 and id_1 respectively. KGC can use two oracles $U(PP, id_\delta, decom_\delta)$ and $U(PP, id_{1-\delta}, decom_{1-\delta})$ to generate sk_δ for id_δ and $sk_{1-\delta}$ for $id_{1-\delta}$.

  1)$sk_\delta = \perp, sk_{1-\delta} = \perp, U$ returns (ζ, ζ) to KGC;
  2)$sk_\delta = \perp, sk_{1-\delta} \neq \perp, U$ returns $(\perp, \zeta)$ to KGC;
  3)$sk_\delta \neq \perp, sk_{1-\delta} = \perp, U$ returns $(\zeta, \perp)$ to KGC;
  4)$sk_\delta \neq \perp, sk_{1-\delta} \neq \perp, U$ returns (sk_0, sk_1) to KGC.
- KGC guesses δ' about δ. In the case that $\delta' = \delta$, KGC wins the game.

Definition 8 (Selective-Failure-Blindness): The PPKeyGen algorithm is selective - failure - blind if KGC is able to win the above game with the negligible advantage of $\epsilon(\lambda)$,

$$Adv_{KGC} = \left|\Pr[\delta' = \delta] - \tfrac{1}{2}\right| < \epsilon(\lambda).$$

Leakage-free. In this game, suppose that the user U is malicious and the KGC is honest, and user U has interaction with KGC in an attempt to get informed about the key. The above game comprises a real-world and an ideal-world scenario, in which a distinguisher $\mathcal{D}$ tries to distinguish the outputs of each scenario.

- **Real-world:** The user U chooses identity θ and interacts with KGC by **PPKeyGen**, and $\mathcal{D}$ can see the interaction between KGC and U.
- **Ideal-world:** The simulator $\mathcal{S}$ chooses an identity θ, then asks a trusted party TP to generate a key through **KeyGen**. $\mathcal{D}$ witnesses the communication process between the simulator $\mathcal{S}$ and the TP.

Definition 9 (Leakage-Freeness): We call that the algorithm is leakage-free if $\mathcal{D}$ can distinguish real-world outputs from ideal-world with only a negligible advantage of $\epsilon(\lambda)$,

$$\begin{aligned} &Adv_U \\ &\quad = \left|\Pr[\mathcal{D}(Real_U^{PPKeyGen}) = 1] - \Pr[\mathcal{D}(Ideal_{\mathcal{S}}^{TP}) = 1]\right| \\ &\quad < \epsilon(\lambda). \end{aligned}$$

Definition 10 (Security of PPTFE-IP): A PPTFE-IP scheme $\Omega = (\mathbf{Setup}, \mathbf{Encrypt}, \mathbf{PPKeyGen}, \mathbf{Decrypt}, \mathbf{Trace})$ is secure if the following two conditions are satisfied:

- The TFE-IP scheme $\Delta = (\mathbf{Setup}, \mathbf{Encrypt}, \mathbf{KeyGen}, \mathbf{Decrypt}, \mathbf{Trace})$ is s-IND-CPA secure;
- The **PPKeyGen** algorithm satisfies two properties: selective-failure-blindness and leakage-freeness.

III. Our Constructions

Firstly, a traceable FE-IP scheme is concretely constructed, then we present the construction of the **PPKeyGen** algorithm.

A. An overview of Our TFE-IP Scheme

High-level overview. Our TFE-IP scheme works as follows: **Firstly**, KGC runs $\mathbf{Setup}$, and generates the master secret key and public parameters. The tracer calculates a secret-public key pair for tracing. **Secondly**, a vector is encrypted by the data owner using the public parameters. **Thirdly**, prior to decryption, a user with the identity θ needs to get a key $sk_{\vec{y},\theta} = \{K_1, K_2, K_3, K_4, K_5\}$ from KGC where K_1 is bound with the vector $\vec{y}$ and K_2 is bound with the identity θ. K_3, K_4 and K_5 are used in tracing and decryption. To prevent collusion attacks, all elements included in a key are bound together by a random number. Users can verify the

Setup. Suppose that $\mathcal{BG}(1^\lambda) \to (e, p, \mathbb{G}, \mathbb{G}_T)$ and $g_0, g_1, g_2 \in \mathbb{G}$ are generators. KGC selects $\vec{s} = (s_1, s_2, \cdots, s_l) \xleftarrow{R} \mathbb{Z}_p^l$ and computes $\vec{h} = \{h_i = g_1^{s_i}\}\ for\ i \in [l]$. Tracer selects $b \xleftarrow{R} \mathbb{Z}_p$ randomly, and calculates $B = g_2^b$. KGC selects $a \xleftarrow{R} \mathbb{Z}_p$ and publishes $Y = g_0^a$. KGC sets $msk = (a, s_i)$ as master secret key, and publishes $(\vec{h}, Y)$. The tracer's secret-public key pair is (b, B). Public parameters of the system can be denoted as $PP = (e, p, \mathbb{G}, \mathbb{G}_T, g_0, g_1, g_2, B, Y, h_1, \cdots, h_l)$.

Encrypt. To encrypt a vector $\vec{x} = (x_1, x_2, \cdots, x_l) \in \mathbb{Z}_p^l$, the data owner first chooses $r \xleftarrow{R} \mathbb{Z}_p$ randomly, then computes

$$ct_i = h_i^r \cdot g_1^{x_i}\ for\ i \in [l], ct_{l+1} = g_1^r, ct_{l+2} = g_2^r, ct_{l+3} = g_0^r.$$

The ciphertext is $Ct = \Big((ct_i)_{i\in[l]}, ct_{l+1}, ct_{l+2}, ct_{l+3}\Big)$. The data owner sends Ct to the cloud server.

KeyGen. Given a vector $\vec{y} = (y_1, y_2, \cdots, y_l)$, to generate a secret key for a user with an identity θ, KGC chooses $w, d \xleftarrow{R} \mathbb{Z}_p$ and computes $sk_{\vec{y},\theta} = \Big\{K_1 = g_0^{\langle\vec{y},\vec{s}\rangle} \cdot B^{\frac{w}{d+a}}, K_2 = \left(g_0 \cdot (g_2 \cdot B)^w \cdot g_2^\theta\right)^{\frac{1}{d+a}}, K_3 = g_1^{\frac{1}{d+a}}, K_4 = w, K_5 = d\Big\}$.

The user receives the key $sk_{\vec{y},\theta}$ and verifies $e(K_1, g_1) = e\left(g_0, \left(\prod_{i=1}^{l} (h_i)^{y_i}\right)\right) \cdot e(B^w, K_3)$; $e\left(K_3, g_0^{K_5} \cdot Y\right) = e(g_0, g_1)$; $e\left(K_2, g_0^{K_5} \cdot Y\right) = e(g_0, g_0) \cdot e(g_0, g_2 \cdot B)^{K_4} \cdot e(g_0, g_2)^\theta$. If all the above equations hold, the secret key $sk_{\vec{y},\theta}$ is valid; otherwise, it is invalid.

Decrypt. After possessing the ciphertext Ct from the cloud server, the user uses his/her secret key $(K_1, K_2, K_3, K_4, K_5)$ to calculate the inner product as follows:

$$e(g_0, g_1)^{\langle\vec{x},\vec{y}\rangle} = \frac{e\left(g_0, \prod_{i=1}^{l} (ct_i)^{y_i}\right) \cdot e(ct_{l+1}, K_2)}{e(K_1, ct_{l+1}) \cdot e(K_3, ct_{l+3}) \cdot e(K_3^{K_4} \cdot K_3^\theta, ct_{l+2})}.$$

Further, the user calculates the discrete logarithm of $e(g_0, g_1)^{\langle\vec{x},\vec{y}\rangle}$ with respect to $e(g_0, g_1)$ to obtain $\langle\vec{x},\vec{y}\rangle$. This discrete logarithmic operation requires that $\langle\vec{x},\vec{y}\rangle$ should not be too large.

Trace. Given a key $sk_{\vec{y},\theta}$, the tracer can compute

$$e(K_3, g_2)^\theta = \frac{e(K_2, g_1)}{e(g_0, K_3) \cdot e(g_2, K_3^{K_4} \cdot K_3^{K_4 \cdot b})}$$

to recover the user's identity associated with the key, where b denotes for tracer's secret key.

Fig. 2. Our TFE-IP Scheme

correctness of his/her secret key. **Fourthly**, the user can use his/her secret key to compute the inner-product of two vectors respectively associated with the ciphertext and the secret key. **Finally**, if tracing is required, only the tracer can use his secret key to recover $e(K_3, g_2)^\theta$ from the secret key $sk_{\vec{y},\theta}$. The tracer computes the discrete logarithm of each identity based on $e(K_3, g_2)$ to discover the identity embedded in the secret key.

B. The Construction of Our TFE-IP Scheme

The formal construction of our TFE-IP scheme is described in Figure 2.

Correctness of our TFE-IP Scheme. The correctness of our TFE-IP scheme is shown by the following equations.

$$
\begin{aligned}
& e\left(g_0, \prod_{i=1}^{l} (ct_i)^{y_i}\right) \cdot e(ct_{l+1}, K_2) \\
&= \left(\prod_{i=1}^{l} e(g_0, h_i^r \cdot g_1^{x_i})^{y_i}\right) \cdot e(g_1^r, \left(g_0 \cdot (g_2 \cdot B)^w \cdot g_2^\theta\right)^{\frac{1}{d+a}}) \\
&= e(g_0, g_1)^{\langle\vec{x},\vec{y}\rangle} \cdot e(g_0, g_1^r)^{\langle\vec{y},\vec{s}\rangle} \cdot e(g_1^{\frac{w}{d+a}}, B^r) \cdot e(g_1^{\frac{1}{d+a}}, g_0^r) \\
&\cdot e(g_1^{\frac{w}{d+a}}, g_2^r) \cdot e(g_1^{\frac{1}{d+a}}, g_2^r)^\theta \\
&= e(g_0, g_1)^{\langle\vec{x},\vec{y}\rangle} \cdot e(K_1, ct_{l+1}) \cdot e(K_3, ct_{l+3}) \\
&\cdot e(K_3, ct_{l+2})^w \cdot e(K_3, ct_{l+2})^\theta \\
&= e(g_0, g_1)^{\langle\vec{x},\vec{y}\rangle} \cdot e(K_1, ct_{l+1}) \cdot e(K_3, ct_{l+3}) \\
&\cdot e(K_3^{K_4} \cdot K_3^\theta, ct_{l+2}),
\end{aligned}
$$

$$K_1 = g_0^{\langle\vec{y},\vec{s}\rangle} \cdot B^{\frac{w}{d+a}}, K_2 = \left(g_0 \cdot (g_2 \cdot B)^w \cdot g_2^\theta\right)^{\frac{1}{d+a}},$$
$$K_3 = g_1^{\frac{1}{d+a}}, K_4 = w, K_5 = d.$$

Therefore,

$$\frac{e\left(g_0, \prod_{i=1}^{l} (ct_i)^{y_i}\right) \cdot e(ct_{l+1}, K_2)}{e(K_1, ct_{l+1}) \cdot e(K_3, ct_{l+3}) \cdot e(K_3^{K_4} \cdot K_3^\theta, ct_{l+2})} = e(g_0, g_1)^{\langle\vec{x},\vec{y}\rangle}.$$

For tracing,

$$
\begin{aligned}
& \frac{e(K_2, g_1)}{e(g_0, K_3) \cdot e(g_2, K_3^{K_4} \cdot K_3^{K_4 \cdot b})} \\
&= \frac{e((g_0 \cdot (g_2 \cdot B)^w \cdot g_2^\theta)^{\frac{1}{d+a}}, g_1)}{e(g_0, g_1^{\frac{1}{d+a}}) \cdot e(g_2, g_1^{\frac{w}{d+a}} \cdot (g_1^{\frac{w}{d+a}})^b)} = e(K_3, g_2)^\theta.
\end{aligned}
$$

PPKeyGen

User		**KGC**
$\left(GID\ \theta, w_1 \overset{R}{\leftarrow} \mathbb{Z}_p, \tau \overset{R}{\leftarrow} \mathbb{Z}_p\right)$		$\left(msk\ \vec{s}, w_2, d \overset{R}{\leftarrow} \mathbb{Z}_p\right)$
1. Select $w_1 \overset{R}{\leftarrow} \mathbb{Z}_p, \tau \overset{R}{\leftarrow} \mathbb{Z}_p$, and compute		
$A_1 = h^{\tau} \cdot B^{w_1}, A_2 = (g_2 \cdot B)^{w_1} \cdot g_2^{\theta}$.		2. Select $w_2 \overset{R}{\leftarrow} \mathbb{Z}_p$ and compute
Generate $\Sigma_U = PoK\left\{(w_1, \theta, \tau) :\right.$	$\xrightarrow[\Sigma_U]{A_1, A_2}$	$B_5 = d, B_1 = \prod_{i=1}^{l} (g_0^{y_i})^{s_i} \cdot (A_1 \cdot B^{w_2})^{\frac{1}{d+a}}$,
$\left. A_1 = h^{\tau} \cdot B^{w_1} \wedge A_2 = (g_2 \cdot B)^{w_1} \cdot g_2^{\theta}\right\}$.		$B_2 = (g_0 \cdot A_2 \cdot (g_2 \cdot B)^{w_2})^{\frac{1}{d+a}}, B_3 = g_1^{\frac{1}{d+a}}, B_4 = h^{\frac{1}{d+a}}$.
3. Compute $w = w_1 + w_2$ and set		Generate $\Sigma_K = PoK\left\{\left(a, w_2, (s_i)_{i\in[l]}\right) :\right.$
$K_1 = \frac{B_1}{B_4^{\tau}}, K_2 = B_2, K_3 = B_3$,	$\xleftarrow[B_4, B_5, \Sigma_K]{w_2, B_1, B_2, B_3}$	$B_2 = (g_0 \cdot A_2 \cdot (g_2 \cdot B)^{w_2})^{\frac{1}{d+a}} \wedge B_1 = g_0^{\langle \vec{y}, \vec{s}\rangle}$
$K_4 = w, K_5 = B_5$.		$\left. \cdot A_1^{\frac{1}{d+a}} \cdot B^{\frac{w_2}{d+a}} \wedge B_3 = g_1^{\frac{1}{d+a}} \wedge B_4 = h^{\frac{1}{d+a}}\right\}$

Fig. 3. Our PPKeyGen Algorithm

C. Our PPKeyGen Algorithm

In order to prevent user collusion attacks, a user's identity is associated with his/her secret key. However, KGC and users may collude to generate new secret keys on behalf of other users. Therefore, we propose a **PPKeyGen** algorithm in Figure 3 to protect user's privacy and prevent KGC's corruption. The user and the KGC collaborate to generate the key in **PPKeyGen** algorithm which employs a zero-knowledge proof and a two-party secure computing protocol, while other users and the KGC cannot learn the embedded identity from the key. The instantiation of zero-knowledge proof Σ_K and Σ_U in our **PPKeyGen** algorithm are presented in Appendix.

Correctness of Our Privacy-Preserving Key Generation Algorithm. Let $w = w_1 + w_2$, the equations presented below demonstrate the correctness of the secret keys generated in Figure 3.

$$\begin{aligned} K_1 &= \frac{B_1}{B_4^{\tau}} = \frac{g_0^{\langle \vec{y}, \vec{s}\rangle} \cdot A_1^{\frac{1}{d+a}} \cdot B^{\frac{w_2}{d+a}}}{\left(h^{\frac{1}{d+a}}\right)^{\tau}} \\ &= g_0^{\langle \vec{y}, \vec{s}\rangle} \cdot B^{\frac{w_1+w_2}{d+a}} = g_0^{\langle \vec{y}, \vec{s}\rangle} \cdot B^{\frac{w}{d+a}}, \\ K_2 &= B_2 = (g_0 \cdot A_2 \cdot (g_2 \cdot B)^{w_2})^{\frac{1}{d+a}} \\ &= \left(g_0 \cdot (g_2 \cdot B)^{(w_1+w_2)} \cdot g_2^{\theta}\right)^{\frac{1}{d+a}} \\ &= \left(g_0 \cdot (g_2 \cdot B)^{w} \cdot g_2^{\theta}\right)^{\frac{1}{d+a}}, \\ K_3 &= g_1^{\frac{1}{d+a}}, K_4 = w_1 + w_2 = w, K_5 = d. \end{aligned}$$

IV. Security Analysis

Theorem 1: Our TFE-IP scheme is $(\epsilon(\lambda), T)$ secure against chosen-plaintext attack (CPA) in the selective model if the DDH assumption holds on the group $\mathbb{G}$ with $(\epsilon(\lambda)', T')$, where $\epsilon(\lambda)' = \frac{\epsilon(\lambda)}{2}$.

Proof: Suppose the existence of an adversary $\mathcal{A}$ who can (t, ϵ)-break the TFE-IP scheme in the IND-CPA security model, there exists a simulator $\mathcal{B}$ who can run $\mathcal{A}$ to break the DDH assumption as below. $\mathcal{C}$ randomly selects a $\mu \in \{0, 1\}$. If $\mu = 0$, $\mathcal{C}$ sends $(g^{\alpha}, g^{\beta}, Z = g^{\alpha\beta})$ to $\mathcal{B}$; if $\mu = 1$, $\mathcal{C}$ sends $(g^{\alpha}, g^{\beta}, Z = g^{\tau})$ to $\mathcal{B}$, in which $\tau \in \mathbb{Z}_p$ is a random number. $\mathcal{B}$ outputs its guess μ' about μ.

Initialization. $\mathcal{A}$ submits two vectors $\vec{x_0} = (x_{0,0}, x_{0,1}, \cdots x_{0,l})$, $\vec{x_1} = (x_{0,0}, x_{0,1}, \cdots, x_{0,l})$, in which l represents the length of vectors.

Setup. $\mathcal{B}$ selects $a, c_0, c_2 \leftarrow \mathbb{Z}_p$, and sets $g_1 = g, g_0 = g^{c_0}, g_2 = g^{c_2}, Y = g_0^{a}$. Let $spc(\vec{x_0} - \vec{x_1})$ be a linear space with basis $(\vec{\eta_1}, \vec{\eta_2}, \cdots, \vec{\eta_l})$ and the basis of $spc(\vec{x_0} - \vec{x_1})^{\perp}$ is $(\vec{\zeta_1}, \vec{\zeta_2}, \cdots, \vec{\zeta_l})$. $\mathcal{B}$ randomly selects a vector $\vec{\pi} = (\pi_1, \pi_2, \cdots, \pi_l) \in spc(\vec{x_0} - \vec{x_1})^{\perp}$, and computes $(h_i = (g_1^{\alpha})^{\pi_i})_{i\in[l]}$. $\mathcal{B}$ returns public parameters $PP = \left\{g_0, g_1, g_2, Y, (h_i)_{i\in[l]}\right\}$ to $\mathcal{A}$. $\mathcal{B}$ impliedly defined $\vec{s} = (s_i)_{i\in[l]} = (\alpha \cdot \pi_i)_{i\in[l]}$.

Phase-I. $\mathcal{A}$ submits $\vec{y} = (y_i)_{i\in[l]} \in \mathbb{Z}_p, \theta \in \mathbb{Z}_p$ with the limitation that $\vec{y} \in spc(\vec{x_0} - \vec{x_1})^{\perp}$. $\mathcal{B}$ selects random $w, d \overset{R}{\leftarrow} \mathbb{Z}_p$ and computes

$$K_1 = g_0^{\langle \vec{y}, \vec{s}\rangle} \cdot B^{\frac{w}{d+a}} = B^{\frac{w}{d+a}}, K_2 = \left(g_0 \cdot (g_2 \cdot B)^{w} \cdot g_2^{\theta}\right)^{\frac{1}{d+a}},$$
$$K_3 = g_1^{\frac{1}{d+a}}, K_4 = w, K_5 = d.$$

$\mathcal{B}$ returns to $\mathcal{A}$ $sk_{\vec{y},U} = \left((K_i)_{i\in[5]}\right)$.

Challenge. $\mathcal{B}$ randomly chooses a $\delta \in \{0, 1\}$. $\mathcal{B}$ calculates

$$ct_i^* = Z^{\pi_i} \cdot g^{x_{\delta,i}}\ for\ i \in [l], ct_{l+1}^* = g^{\beta},$$
$$ct_{l+2}^* = g^{\beta^{c_2}}, ct_{l+3}^* = g^{\beta^{c_0}},$$

outputs ciphertext $Ct^* = \left((ct_i^*)_{i\in[l]}, ct_{l+1}^*, ct_{l+2}^*, ct_{l+3}^*\right)$.

Phase-II. Repeat the same process as in **Phase-I**.

Output. $\mathcal{A}$ guesses δ' about δ. If $\delta' \neq \delta, \mathcal{B}$ produces $\mu' = 1$ as output; If $\delta' = \delta, \mathcal{B}$ produces $\mu' = 0$ as output.

The remaining thing to complete the proof is to calculate the advantage with which $\mathcal{B}$ can break the DDH assumption.

- If $\mu = 0, Z = g^{\alpha\beta}, ct_i^* = Z^{\pi_i} \cdot g^{x_{\delta,i}} = g^{\alpha\beta\pi_i} \cdot g^{x_{\delta,i}}, ct_{l+1}^* = g^{\beta}, ct_{l+2}^* = g^{\beta^{c_2}}, ct_{l+3}^* = g^{\beta^{c_0}}$ is a correct ciphertext of $\vec{x_\delta}$. Therefore, $\Pr[\delta' = \delta | \mu = 0] = \frac{1}{2} + \epsilon(\lambda)$. When $\delta' = \delta, \mathcal{B}$ outputs $\mu' = 0$, so $\Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \epsilon(\lambda)$.
- If $\mu = 1, Z = g^{\tau}, ct_i^* = Z^{\pi_i} \cdot g^{x_{\delta,i}} = g^{\tau\pi_i} \cdot g^{x_{\delta,i}}, ct_{l+1}^* = g^{\beta}, ct_{l+2}^* = g^{\beta^{c_2}}, ct_{l+3}^* = g^{\beta^{c_0}}$. This information theoretically hide both $\vec{x_0}$ and $\vec{x_1}$. Therefore,

$\Pr[\delta' \neq \delta | \mu = 1] = \frac{1}{2}$. When $\delta' \neq \delta, \mathcal{B}$ produces $\mu' = 1$ as output, so $\Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$.

In conclusion, the advantage of breaking the DDH assumption by $\mathcal{B}$ can be computed as follows:

$$\begin{aligned}&\left|\tfrac{1}{2}\Pr[\mu' = \mu|\mu = 0] - \tfrac{1}{2}\Pr[\mu' \neq \mu|\mu = 1]\right| \\ &= \tfrac{1}{2}\cdot\left(\tfrac{1}{2} + \epsilon(\lambda)\right) - \tfrac{1}{2}\cdot\tfrac{1}{2} = \tfrac{\epsilon(\lambda)}{2}.\end{aligned}$$

A. Traceability

Theorem 2: The proposed TFE-IP scheme is $(\epsilon(\lambda), T) - traceable$ if the q-SDH assumption holds with an advantage no more than $\epsilon_1(\lambda)$ and a running time T_1, and the DL assumption holds with an advantage no greater than $\epsilon_2(\lambda)$ and a running time T_2, where $\epsilon(\lambda) = \max\left\{\frac{\epsilon_1(\lambda)}{4}\times\left(1-\frac{q}{p}\right)+\frac{\epsilon_2(\lambda)}{4}\times\frac{p-1}{p^3}, \frac{\epsilon_1(\lambda)}{4}\times\left(\left(1-\frac{q}{p}\right)+\frac{1}{q}\right)\right\}$, q represents the number of queries made by $\mathcal{A}$ and $T = T_1 + T_2 + O(T_1) + O(T_2)$.

Proof: Suppose the existence of an adversary $\mathcal{A}$ who can break the traceability of our scheme, there is a simulator $\mathcal{B}$ who is able to run $\mathcal{A}$ to solve the q-SDH problem as follows. $\mathcal{C}$ sends $\left(g, g^z, g^{z^2}, \cdots, g^{z^q}, h, h^z\right)$ to $\mathcal{B}$. $\mathcal{B}$ will produce (c, g^{z+c}) as output where $c \in \mathbb{Z}_p$.

Setup. $\mathcal{B}$ sets $(\Phi_i = g^{z_i})_{i\in[q]}$, randomly selects $a, \mu_1, \mu_2, \cdots \mu_{q-1} \xleftarrow{R} \mathbb{Z}_p$, $\vec{s} = (s_1, s_2, \cdots, s_l) \xleftarrow{R} \mathbb{Z}_p^l$, and lets $f(z) = \prod_{i=1}^{q-1}(z+\mu_i) = \Sigma_{i=0}^{q-1}\delta_i z^i$, then computes $\tilde{g} = \prod_{i=0}^{q-1}(g^{z_i})^{\delta_i} = g^{f(z)}, \hat{g} = \prod_{i=0}^{q-1}(g^{z^{i+1}})^{\delta_i} = \tilde{g}^z, \vec{h} = \{h_i = g_1^{s_i}\}\ for\ i \in [l], Y = g_0^a$. $\mathcal{B}$ chooses $\gamma_1, \gamma_2, \gamma_3, \pi, \rho, \kappa, b \xleftarrow{R} \mathbb{Z}_p$, sets $B = \tilde{g}^b$, then calculates $g_0 = \tilde{g}^{\gamma_1}, g_2 = \left((\hat{g}\tilde{g}^\pi)^\kappa\,\tilde{g}^{-1}\right)^{\frac{1}{\rho}} = \tilde{g}^{\frac{(z+\pi)\kappa-1}{\rho}}, \tilde{g} = g^{\gamma_3}$. B is the trace public key. $\mathcal{A}$ is given public parameters $(e, p, \tilde{g}, \hat{g}, g_0, g_1, g_2, g_3, B, Y, h_1, \cdots, h_l)$ from $\mathcal{B}$.

Key Query. $\mathcal{A}$ sends $(\theta_i, \vec{y_i})_{i\leq q}$ to $\mathcal{B}$ for the $i-th$ query. $\mathcal{B}$ sets $f_i(z) = \frac{f(z)}{z+\mu_i} = \Sigma_{j=0}^{q-2}\eta_{i_j}z^j$, randomly selects $w \xleftarrow{R} \mathbb{Z}_p$, computes

$$\begin{aligned}\nu_i &= \prod_{j=0}^{q-2}(\Phi_j)^{\eta_{i_j}} = g^{f_i(z)} = g^{\frac{f(z)}{z+\mu_i}} = \tilde{g}^{\frac{1}{z+\mu_i}},\\ K_1 &= g_0^{\langle\vec{y},\vec{s}\rangle}\cdot\nu_i^{b\cdot w}, K_3 = \nu_i^{\gamma_1}, K_4 = w, K_5 = \mu_i,\\ K_2 &= \prod_{j=0}^{q-2}\left(g^{z^{j+1}}\right)^{\eta_{i_j}\cdot\frac{\kappa}{\rho}\cdot(\gamma_2\cdot w+\theta_i)}\\ &\cdot\prod_{j=0}^{q-2}\left(g^{z^j}\right)^{\eta_{i_j}\cdot\left(\left(\frac{(\pi\cdot\kappa-1)}{\rho}\right)\cdot(\gamma_2\cdot w+\theta_i)+\gamma_1\right)},\end{aligned}$$

and returns $sk_{(\theta_i,\vec{y_i})} = (K_1, K_2, K_3, K_4, K_5)$ to $\mathcal{A}$. We prove that $sk_{(\theta_i,\vec{y_i})}$ is correct.

$$\begin{aligned}K_1 &= g_0^{\langle\vec{y},\vec{s}\rangle}\cdot\nu_i^{b\cdot w} = g_0^{\langle\vec{y},\vec{s}\rangle}\cdot\tilde{g}^{\frac{b\cdot w}{z+\mu_i}} = g_0^{\langle\vec{y},\vec{s}\rangle}\cdot B^{\frac{w}{z+\mu_i}}\\ K_2 &= \prod_{j=0}^{q-2}\left(g^{z^{j+1}}\right)^{\eta_{i_j}\cdot\frac{\kappa}{\rho}\cdot(\gamma_2\cdot w+\theta_i)}\\ &\cdot\prod_{j=0}^{q-2}\left(g^{z^j}\right)^{\eta_{i_j}\cdot\left(\left(\frac{(\pi\cdot\kappa-1)}{\rho}\right)\cdot(\gamma_2\cdot w+\theta_i)+\gamma_1\right)}\\ &= g^{(\Sigma_{j=0}^{q-2}\eta_{i_j}\cdot z^{j+1})\cdot\frac{\kappa}{\rho}\cdot(\gamma_2\cdot w+\theta_i)}\\ &\cdot g^{(\Sigma_{j=0}^{q-2}\eta_{i_j}\cdot z^j)\cdot\left(\left(\frac{(\pi\cdot\kappa-1)}{\rho}\right)\cdot(\gamma_2\cdot w+\theta_i)+\gamma_1\right)}\\ &= g^{\gamma_1\cdot f_i(z)}\cdot g^{f_i(z)\cdot\frac{(z+\pi)\kappa-1}{\rho}\cdot(\gamma_2\cdot w+\theta_i)}\\ &= \tilde{g}^{\frac{\gamma_1}{z+\mu_i}}\cdot\tilde{g}^{\frac{(z+\pi)\kappa-1}{\rho}\cdot\frac{\gamma_2\cdot w+\theta_i}{z+\mu_i}}\\ &= g_0^{\frac{1}{z+\mu_i}}\cdot g_2^{\frac{\gamma_2\cdot w+\theta_i}{z+\mu_i}} = (g_0\cdot(g_2\cdot B)^w\cdot g_2^{\theta_i})^{\frac{1}{z+\mu_i}},\\ K_3 &= g_1^{\frac{1}{z+\mu_i}} = \tilde{g}^{\frac{\gamma_1}{z+\mu_i}} = \nu_i^{\gamma_1}.\end{aligned}$$

Trace Query. $\mathcal{A}$ adaptively submits a key $sk_i = (K_1, K_2, K_3, K_4, K_5)$ to $\mathcal{B}$. $\mathcal{B}$ computes

$$\frac{e(K_2, g_1)}{e(g_0, K_3)\cdot e(g_2, K_3^{K_4}\cdot K_3^{K_4\cdot b})} = e\left(K_3, g_2\right)^\theta.$$

$\mathcal{B}$ sends to $\mathcal{A}$ the discrete logarithm of identity θ.

Key Forgery. $\mathcal{A}$ outputs $sk_{(\theta^*,\vec{y}^*)} = (K_1^*, K_2^*, K_3^*, K_4^*, K_5^*)$ and sends the key to $\mathcal{B}$. Let's consider the two types of forgery as follows: **Type I.** The identity associated with the key was not previously queried, namely, $\theta^* \notin \{\theta_1, \theta_2, \cdots\theta_q\}$. Furthermore, this forgery is divided into two cases:

$\boldsymbol{Case-I:\ \theta^* \notin \{\theta_1, \theta_2, \cdots\theta_q\}, \mu^* \notin \{\mu_1, \mu_2, \cdots\mu_{q-1}, \pi\}}$.

Set $f(z)_1^* = \frac{f(z)}{z+\mu_i} = \Sigma_{j=0}^{q-2}\psi_j z^j, f(z)_2^* = \frac{f(z)\cdot(z+\pi)}{z+\mu_i} = \Sigma_{j=0}^{q-1}\psi'_j z^j, f(z) = (z+\mu^*)\cdot\sigma(z)+\chi, \sigma(z) = \Sigma_{j=0}^{q-2}\sigma_j\cdot z^j$ and thus $K_2^* = (g_0\cdot(g_2\cdot B)^{K_4^*}\cdot g_2^{\theta^*})^{\frac{1}{z+K_5^*}} = (g_0\cdot(g_2\cdot B)^{w^*}\cdot g_2^{\theta^*})^{\frac{1}{z+\mu^*}} = (g_0^{\frac{1}{z+\mu^*}})\cdot((g_2\cdot B)^{w^*}\cdot g_2^{\theta^*})^{\frac{1}{z+\mu^*}}$.

Furthermore, we have

$$\begin{aligned}g_0^{\frac{1}{z+\mu^*}} &= K_2^*\cdot((g_2\cdot B)^{w^*}\cdot g_2^{\theta^*})^{\frac{-1}{z+\mu^*}}\\ &= K_2^*\cdot((\tilde{g}^{\frac{((z+\pi)\kappa-1)\gamma_2 w^*}{\rho}})\cdot\tilde{g}^{\frac{((z+\pi)\kappa-1)\theta^*}{\rho}})^{\frac{-1}{z+\mu^*}}\\ &= K_2^*\cdot\tilde{g}^{\frac{-((z+\pi)\kappa)(\gamma_2\cdot w^*+\theta^*)}{\rho(z+\mu^*)}}\cdot\tilde{g}^{\frac{\gamma_2 w^*+\theta^*}{\rho(z+\mu^*)}}\\ &= K_2^*\cdot g^{\frac{-f(z)((z+\pi)\kappa)(\gamma_2\cdot w^*+\theta^*)}{\rho\cdot(z+\mu^*)}}\cdot g^{\frac{f(z)\cdot(\gamma_2\cdot w^*+\theta^*)}{\rho\cdot(z+\mu^*)}}\\ &= K_2^*\cdot g^{\frac{-f_2^*(z)\cdot\kappa)\cdot(\gamma_2\cdot w^*+\theta^*)}{\rho}}\cdot g^{\frac{f_1^*(z)\cdot(\gamma_2\cdot w^*+\theta^*)}{\rho}}\\ &= K_2^*\cdot\prod_0^{q-1}\left(g^{z^i}\right)^{\frac{-\psi'_i\cdot\kappa\cdot(\gamma_2\cdot w^*+\theta^*)}{\rho}}\cdot\prod_0^{q-2}\left(g^{z^i}\right)^{\frac{\psi_i\cdot(\gamma_2\cdot w^*+\theta^*)}{\rho}}.\end{aligned}$$

Set $\Theta = K_2^*\cdot\prod_0^{q-1}\left(g^{z^i}\right)^{\frac{-\psi'_i\cdot\kappa\cdot(\gamma_2\cdot w^*+\theta^*)}{\rho}}\cdot\prod_0^{q-2}\left(g^{z^i}\right)^{\frac{\psi_i\cdot(\gamma_2\cdot w^*+\theta^*)}{\rho}}$, we have $\Theta = g_0^{\frac{1}{z+\mu^*}} = g^{\frac{f(z)\cdot\gamma_1}{z+\mu^*}} = g^{\gamma_1\frac{(z+\mu^*)\cdot\sigma(z)+\chi}{z+\mu^*}} = g^{\gamma_1\sigma(z)}\cdot g^{\gamma_1\frac{\chi}{z+\mu^*}}$. Therefore, we have

$$\begin{aligned}g^{\frac{1}{z+\mu^*}} &= \left(\Theta\cdot g^{-\sigma(z)}\right)^{\frac{1}{\chi\cdot\gamma_1}}\\ &= \left(K_2^*\cdot\prod_{i=0}^{q-1}\left(g^{z^i}\right)^{\frac{-\psi'_i\kappa(\gamma_2 w^*+\theta^*)}{\rho}}\right.\\ &\left.\cdot\prod_{j=0}^{q-2}\left(g^{z^j}\right)^{\frac{\psi_j\cdot(\gamma_2 w^*+\theta^*)}{\rho}}\cdot\prod_{k=0}^{q-2}\left(g^{z^k}\right)^{(-\sigma_k)}\right)^{\frac{1}{\chi\gamma_1}}.\end{aligned}$$

$\mathcal{B}$ can output $(\mu^*, g^{\frac{1}{z+\mu^*}})$ from the above generation. Therefore, $\mathcal{B}$ is able to utilize $\mathcal{A}$ to solve the $q-SDH$ problem. The probability that $\mu^* \notin \{\mu_1, \mu_2, \cdots, \mu_{q-1}, \mu\}$ is $(1-\frac{q}{p})$.

$\boldsymbol{Case-II:\ \theta^* \notin \{\theta_1, \theta_2, ..\theta_q\}, \mu^* = \mu_i}$.

In this case, we have $K_2^* = (g_0\cdot(g_2\cdot B)^{w^*}\cdot g_2^{\theta^*})^{\frac{1}{z+\mu^*}}, (K_2)_i =$

$(g_0 \cdot (g_2 \cdot B)^{w_i} \cdot g_2^{\theta_i})^{\frac{1}{z+\mu_i}}$. Given $\mu^* = \mu_i, K_2^* = (K_2)_i$, we obtain

$$g_0 \cdot (g_2 \cdot B)^{w^*} \cdot g_2^{\theta^*} = g_0 \cdot (g_2 \cdot B)^{w_i} \cdot g_2^{\theta_i}.$$

$\mathcal{B}$ can compute $\log_{g_2} B = \frac{\theta_i - \theta^* - w^* + w_i}{w^* - w_i}$ by using $\mathcal{A}$. Therefore, $\mathcal{B}$ is able to break the DL assumption by using $\mathcal{A}$. The probability of this case can be computed as $\frac{1}{p} \cdot \frac{1}{p} \cdot (1 - \frac{1}{p}) = \frac{p-1}{p^3}$.

Type-II. The user's identity related to the key has been previously queried, namely, $\theta^* \in \{\theta_1, \theta_2, ..\theta_q\}$. We take the two cases as follows into consideration.

$\boldsymbol{Case - III : \theta^* \in \{\theta_1, \theta_2, ..\theta_q\}, \mu^* \notin \{\mu_1, \mu_2, ..\mu_{q-1}, \pi\}}$.

$\mathcal{B}$ can compute $(\mu, g^{\frac{1}{z+\mu}})$ in the same way as *Case-I* to break the $q - SDH$ assumption with the probability that $\mu^* \notin \{\mu_1, \mu_2, ..\mu_{q-1}, \mu\}$ is $(1 - \frac{q}{p})$.

$\boldsymbol{Case - IV : \theta^* \in \{\theta_1, \theta_2, ..\theta_q\}, \mu^* \in \{\mu_1, \mu_2, ..\mu_{q-1}, \pi\}, K_2^* = (K_2)_i}$.

The probability that $\mu^* = \mu$ is $\frac{1}{q}$. Because $\mu \notin \{\mu_1, \mu_2, ..\mu_{q-1}\}$, $\mathcal{B}$ can compute $(\mu, g^{\frac{1}{z+\mu}})$ in the same way as *Case-I* to solve the $q - SDH$ problem.

For the sake of completeness of the proof, we need to analyse the advantage $\mathcal{B}$ posses in solving the $q - SDH$ problem. The probabilities of *Case-I*, *Case-II*, *Case-III* and *Case-IV* forgeries are denoted as $\Pr[Case - I]$, $\Pr[Case - II]$, $\Pr[Case - III]$ and $\Pr[Case - IV]$ respectively. The four cases are independently and identically distributed, each occurring with a probability of $\frac{1}{4}$. Hence, the advantage of breaking the $q - SDH$ assumption and DL assumption by $\mathcal{B}$ can be calculated as follows.

$$\begin{aligned}
&\Pr[\text{Type-I}] = \Pr[Case - I] + \Pr[Case - II] \\
&= \frac{\epsilon_1(\lambda)}{4}\left(1 - \frac{q}{p}\right) + \frac{\epsilon_2(\lambda)}{4}\frac{p-1}{p^3}, \\
&\Pr[\text{Type-II}] = \Pr[Case - III] + \Pr[Case - IV] \\
&= \frac{\epsilon_1(\lambda)}{4}\left(\left(1 - \frac{q}{p}\right) + \frac{1}{q}\right), \\
&\epsilon(\lambda) = \max\{\Pr[\text{Type-I}], \Pr[\text{Type-II}]\} \\
&= \max\left\{\frac{\epsilon_1(\lambda)}{4} \times \left(1 - \frac{q}{p}\right) + \frac{\epsilon_2(\lambda)}{4} \times \frac{p-1}{p^3},\right. \\
&\left.\frac{\epsilon_1(\lambda)}{4} \times \left(\left(1 - \frac{q}{p}\right) + \frac{1}{q}\right)\right\}.
\end{aligned}$$

B. Privacy Preservation

Theorem 3: The PPKeyGen algorithm depicted in Figure 3 satisfies both $\mathbf{leakage - freeness}$ and **selective-failure-blindness** under the DL assumption.

Lemma 1: Under the DL assumption, the PPKeyGen algorithm explicated in Figure 3 is selective-failure blindness.

Proof: KGC is malicious and tries to distinguish the user's identity θ embedded in the key. Suppose that g_0, g_1, g_2 are generators in group $\mathbb{G}$, $\mathcal{BG}(1^\lambda) \to (e, p, \mathbb{G}, \mathbb{G}_T)$. KGC selects $\vec{s} = (s_1, s_2, \cdots, s_l) \xleftarrow{R} \mathbb{Z}_p^l$ and computes $\vec{h} = \{h_i = g_1^{s_i}\}\ for\ i \in [l]$.Tracer selects $b \xleftarrow{R} \mathbb{Z}_p$ randomly and calculates $B = g_2^b$. KGC selects a random a and publishes $Y = g_0^a$. KGC sets $msk = (a, s_i)$ as the master secret key and publishes $(\vec{h}, Y)$. The tracer's secret-public key pair is (b, B). Public parameters of the system can be denoted as $PP = (e, p, \mathbb{G}, \mathbb{G}_T, g_0, g_1, g_2, B,\ Y, h_1, \cdots, h_l)$. KGC submits $(\theta_0, \vec{y})$ and $(\theta_1, \vec{y})$ and selects $\delta \in \{0, 1\}$ randomly. KGC can utilize the user oracles $U(PP, \theta_\delta, decom_\delta)$ and $U(PP, \theta_{1-\delta},\ decom_{1-\delta})$. Subsequently, KGC and an honest user U perform the protocol illustrated in Figure 3. The oracle U will output $sk_0(\theta_0, \vec{y})$ and $sk_1(\theta_1, \vec{y})$.

$$\begin{aligned}
&1) sk_\delta = \perp, sk_{1-\delta} = \perp, \text{U returns } (\zeta, \zeta) \text{ to } KGC; \\
&2) sk_\delta = \perp, sk_{1-\delta} \neq \perp, \text{U returns } (\perp, \zeta) \text{ to } KGC; \\
&3) sk_\delta \neq \perp, sk_{1-\delta} = \perp, \text{U returns } (\zeta, \perp) \text{ to } KGC; \\
&4) sk_\delta \neq \perp, sk_{1-\delta} \neq \perp, \text{U returns } (sk_0, sk_1) \text{ to } KGC.
\end{aligned}$$

In PPKeyGen, the user computes A_1, A_2, generates $\Sigma_U = PoK\{(w_1, \theta, \tau) : A_1 = h^\tau \cdot B^{w_1} \wedge A_2 = (g_2 \cdot B)^{w_1} \cdot g_2^\theta\}$ and sends A_1, A_2, Σ_U to KGC. Until this stage, KGC executes either one or both oracles whose perspective on the two oracles remains computational indistinguishable, because of the hiding property of commitment schemes and the zero-knowledge property of zero-knowledge proofs. When KGC has the ability to compute the $k = (K_1, K_2, K_3, K_4, K_5)$ for the first oracle, it can predict k_δ without using the steps below.

- Firstly, KGC verifies $\Sigma_K = PoK\{\Big(a, (s_i)_{i\in[l]}\Big) : B_1 = g_0^{\langle \vec{y}, \vec{s} \rangle} \cdot A_1^{\frac{1}{d+a}} \cdot B^{\frac{w_2}{d+a}} \wedge B_2 = g_0^{\frac{1}{d+a}} \cdot A_2^{\frac{1}{d+a}} \cdot (g_2 \cdot B)^{\frac{w_2}{d+a}} \wedge B_3 = g_1^{\frac{1}{d+a}} \wedge e(B_3, g_1^{B_5} \cdot Y) = e(g_1, g_1) \wedge e(B_4, g_1^{B_5} \cdot Y) = e(g_1, h)\}$. If it is invalid, KGC returns $k_0 = \perp$.
- For the second oracle, KGC outputs another $\chi = (B_1, B_2, B_3, B_4, B_5)$ and generates $\Sigma_K = PoK\{\Big(a, (s_i)_{i\in[l]}\Big) : B_1 = g_0^{\langle \vec{y}, \vec{s} \rangle} \cdot A_1^{\frac{1}{d+a}} \cdot B^{\frac{w_2}{d+a}} \wedge B_2 = g_0^{\frac{1}{d+a}} \cdot A_2^{\frac{1}{d+a}} \cdot (g_2 \cdot B)^{\frac{w_2}{d+a}} \wedge B_3 = g_1^{\frac{1}{d+a}} \wedge e(B_3, g_1^{B_5} \cdot Y) = e(g_1, g_1) \wedge e(B_4, g_1^{B_5} \cdot Y) = e(g_1, h)\}$. If it is invalid, KGC returns $k_1 = \perp$.
- If both of the two steps above are successful, then KGC proceeds as follows:

$$\begin{aligned}
&1) k_0 = \perp, k_1 = \perp, \text{U returns } (\zeta, \zeta) \text{ to KGC;} \\
&2) k_0 = \perp, k_1 \neq \perp, \text{U returns } (\perp, \zeta) \text{ to KGC;} \\
&3) k_0 \neq \perp, k_1 = \perp, \text{U returns } (\zeta, \perp) \text{ to KGC.}
\end{aligned}$$

- If $k_0 \neq \perp, k_1 \neq \perp$, KGC returns θ_0 and θ_1. KGC aborts when any of them fails, otherwise it returns (k_0, k_1).

The prediction of $sk_0(\theta_0, \vec{y})$ and $sk_1(\theta_1, \vec{y})$ is right and consistent with the oracle's distribution. Consequently, if both proofs are successful, by implementing $PPKeyGen$(KGC$\leftrightarrow$U), KGC can generate a valid secret key that U possesses. Therefore, if KGC can forecast the output of $U(PP, \theta_\delta, decom_\delta)$ and $U(PP, \theta_{1-\delta}, decom_{1-\delta})$, KGC's advantage in distinguishing between the two oracles is the same as the probability of no interaction. Therefore, the advantage of KGC should come from the received A_1, A_2 and the proof $\Sigma_U = PoK\{(w_1, \theta, \tau) : A_1 = h^\tau \cdot B^{w_1} \wedge A_2 = (g_2 \cdot B)^{w_1} \cdot g_2^\theta\}$. Due to the witness indistinguishability property of zero-knowledge proofs and the hiding property of commitment schemes, KGC is incapable of distinguishing between $U(PP, \theta_\delta, decom_\delta)$ and $U(PP, \theta_{1-\delta}, decom_{1-\delta})$ with a non-negligible advantage.

TABLE III
COMMUNICATION COST COMPARISON BETWEEN EXISTING WORK AND OUR TFE-IP SCHEME IN FIGURE 2

Scheme	Setup	Key Generation			Encryption
		KeyGen	PPKeyGen		
[29]	$(2\ell+1)\lvert\mathbb{Z}_p\rvert+(\ell+1)\lvert\mathbb{G}_1\rvert+(\ell+1)\lvert\mathbb{G}_T\rvert$	$\lvert\mathbb{G}_2\rvert$	–		$\ell\lvert\mathbb{G}_T\rvert+\ell\lvert\mathbb{G}_1\rvert$
[30][1]	$(\ell+2)\lvert\mathbb{G}\rvert+2\ell\lvert\mathbb{Z}_p\rvert$	$\ell\lvert\mathbb{S}\rvert+2\lvert\mathbb{Z}_p\rvert$	–		$\ell\lvert\mathbb{Z}_p\rvert+(\ell+2)\lvert\mathbb{G}\rvert+\ell\lvert\mathbb{S}\rvert$
Ours	$(\ell+5)\lvert\mathbb{G}_1\rvert+(\ell+2)\lvert\mathbb{Z}_p\rvert$	$2\lvert\mathbb{Z}_p\rvert+3\lvert\mathbb{G}_1\rvert$	User: $5\lvert\mathbb{G}_1\rvert+11\lvert\mathbb{Z}_p\rvert$	KGC: $(4\ell+5)\lvert\mathbb{Z}_p\rvert+(\ell+8)\lvert\mathbb{G}_1\rvert$	$(\ell+3)\lvert\mathbb{G}_1\rvert+\lvert\mathbb{Z}_p\rvert$

1. In [30], $|\mathbb{S}|$ means the size of one element in the public directory pd, namely, a vector of size ℓ.

TABLE IV
COMPUTATION COST COMPARISON BETWEEN EXISTING WORK AND OUR TFE-IP SCHEME IN FIGURE 2

Scheme	Setup	Encryption	Key Generation			Decryption	Trace
			KeyGen	PPKeyGen			
[29][1]	$\ell E_{x_1}+\ell E_{x_T}+P$	$\ell E_{x_1}+2\ell E_{x_T}$	E_{x_2}	–		$\ell E_{x_1}+\ell E_{x_T}+P$	$\frac{8\lambda N^2}{\mu(\lambda)}\cdot(\ell E_{x_1}+2\ell E_{x_T})$
[30][1]	$2\ell E_x$	$(2\ell+2)E_x$	0	–		$(\ell+3)E_x$	$\frac{(N+1)\lambda N^2}{\mu(\lambda)}\cdot(2\ell+2)E_x$
Ours[2]	$(\ell+2)E_x$	$(2\ell+3)E_x$	$9P+(\ell+11)E_x$	User: $(3\ell+25)E_x+2E_H$	KGC: $(\ell+18)E_x+2E_H$	$(\ell+2)E_x+5P$	$4P+3E_x$

1. In [29], [30], N means user number, λ represents security parameter and $\mu(\lambda)$ is a non-negligible function of λ .
2. This cost includes an extra cost of Key Verification, while the other schemes don't include the Key Verification function.

Lemma 2: The PPKeyGen algorithm, as depicted in Figure 3, is leakage-free.

Proof: Assuming the existence of a malicious user U in the real-world experiment, U interacts with an honest KGC executing the **PPKeyGen** protocol. A corresponding simulator $\mathcal{S}$ can be established in the ideal experimental, which has access to the honest KGC performing **KeyGen** algorithm in ideal world. $\mathcal{S}$ conveys the input of $\mathcal{D}$ to U and U's output to $\mathcal{D}$, simulating the interaction between $\mathcal{D}$ and U. Process of the real-world experiment is shown below.

- The simulator $\mathcal{S}$ sends public parameters PP to malicious user U. U computes $A=(A_1,A_2)$ and generates $\Sigma_U = PoK\{(w_1,\theta,\tau) : A_1 = h^{\tau}\cdot B^{w_1} \wedge A_2 = (g_2\cdot B)^{w_1}\cdot g_2^{\theta}\}$. $\mathcal{S}$ aborts if the proof fails. Otherwise, $\mathcal{S}$ can rewind (w_1,θ,τ) from Σ_U.
- $\mathcal{S}$ sends θ to the honest KGC, and executes **KeyGen** to generate (K_1,K_2,K_3,K_4,K_5). $\mathcal{S}$ computes $B_4 = K_4, B_1 = K_1\cdot B_4^{\tau}, B_2 = K_2, B_3 = K_3, B_5 = K_5$.

We assume (K_1,K_2,K_3,K_4,K_5) is a valid key generated by the honest KGC under the ideal-world experiment, while (B_1,B_2,B_3,B_4,B_5) is a valid key from KGC under the real-world experiment. Furthermore, (B_1,B_2,B_3,B_4,B_5) distributes identically, and so is (K_1,K_2,K_3,K_4,K_5). Therefore, $\mathcal{D}$ cannot distinguish the real-world experiment from the ideal-world experiment.

V. COMPARISON AND IMPLEMENTATION

A. Comparison and Efficiency Analysis

We conduct a performance comparison, including communication cost and computation cost of our PPTFE-IP scheme in Table III and Table IV with existing schemes [29], [30], [31], [33] in this subsection. We mainly consider the computationally intensive operations such as exponential, pairing and hash operations, while disregarding other operations. ℓ represents the dimension of vectors; $|\mathbb{G}_1|$, $|\mathbb{G}_T|$ and $|\mathbb{Z}_p|$ represent the length of an element on group $\mathbb{G}$, $\mathbb{G}_T$ and $\mathbb{Z}_p$ respectively; E_x and E_{x_T} represent the cost of performing one exponential operation on $\mathbb{G}$ and $\mathbb{G}_T$; E_H represents the time of performing one hash operation, and P stands for the time of performing one pairing operation.

Table III presents the communication costs of the three schemes: [29], *Trace-and-Revoke FE-IP scheme under DDH* described in Section 5.2 of [30] and ours. Table IV shows computation costs of the three schemes. We first construct a TFE-IP scheme, and then propose a privacy-preserving key generation algorithm. In Table III and Table IV, we compare the complexity of the traceable module with that of existing TFE-IP schemes [29], [30]. From Table III, with the assumption that $l \gg 1$, we know that the communication cost of our **Setup** and **Encryption** is lower than the other two schemes, while **Key Generation** is slightly more expensive than [29], [30]. Moreover, as observed in Table IV, the computation cost of our traceable FE-IP scheme is about the same as [29], [30]. Since [29] and [30] are TFE-IP, we compare our TFE-IP scheme with them in Table III and IV. As shown in the tables, the computation cost and communication cost of all algorithms in our PPTFE-IP scheme are linear with the dimension ℓ, except the **Trace algorithm**.

B. Implementation and Evaluation

Our TFE-IP scheme introduced in Figure 2, the PPKeyGen algorithm depicted in Figure 3 are implemented and evaluated. We conduct the performance measurements on a computer running Ubuntu 20.04 (64 bit) system with an Intel (R) Core (TM) i7-8550U CPU and 8G of RAM. The PBC library [42] in Linux is utilized to perform bilinear pairing operations. The end-to-end communication is simulated by using Socket programming in C language on Linux [43]. In our network setting, the bandwidth of our network is 909.68MB/s, the minimum value of RTT (Round Trip Time) is 0.025ms, the average value is 0.036ms, the maximum value is 0.049ms and the mean deviation value is 0.006ms.

In practice, the vector length is limited by the efficiency of the encryption computation, and it is recommended to use vec-

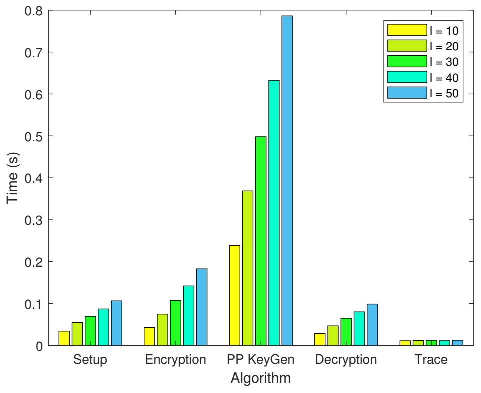


Fig. 4. The Computation Cost of Our PPTFE-IP Scheme

tors with lengths between 10-100. In federated learning[10], local lightweight models on edge devices can be updated by computing gradients via FE-IP, with parameter vectors of dimensions between 10-100. In IoT[13], the crowdsourcing platform efficiently implements the nearest task assignments without revealing sensitive information about tasks and workers using FE-IP in the server assignment model, and the recommended vector size is 10-20. In cloud computing[41], FE-IP can be used to compute the inner product of encrypted feature vectors on the cloud to support privacy-preserving similarity joins (e.g., medical record matching), the vector length is typically set to 10-50.

When implementing our scheme, for simplicity, we consider the five dimensions: $\ell = 10, \ell = 20, \ell = 30, \ell = 40$ and $\ell = 50$. Each time is obtained by taking the average value after 10 experiments. Figure 4 depicts the time spent by the algorithms in each stage of our PPTFE-IP scheme. The **Setup** algorithm takes 34.199ms, 54.722ms, 69.529ms, 87.157ms and 106.472ms to setup the system in the above five cases. The **Encryption** algorithm costs 42.920ms, 74.868ms, 107.548ms, 142.085ms and 182.863ms for each case, which takes more time than **Setup**. Due to the two- party secure computing, the **PPKeyGen** algorithm takes a longer execution time than the other algorithms. It takes 238.759ms, 368.7528ms, 497.998ms, 632.433ms and 786.247ms to generate a secret key in the above five cases. Moreover, multiple pairing and exponentiation operations are required during the key verification, which is time-consuming. The **Decryption** algorithm costs 28.677ms, 46.864ms, 64.910ms, 80.270ms and 98.845ms respectively. The **Trace** algorithm takes 11.307ms, 11.991ms, 12.283ms, 11.405ms and 12.432ms in the five cases, respectively. It is indicated that our **Trace** algorithm is efficient since it is independent of the vector dimension.

VI. Conclusion and Future Work

Cloud computing is an emerging computing paradigm in which the resources of computing infrastructure are provided as a service over the Internet. When users outsource sensitive data to cloud servers for sharing, traditional public-key encryption protects data confidentiality in cloud computing, while functional encryption provides a more fine-grained decryption method. However, in functional encryption, the user may collude with malicious users to sell private keys. To protect users' privacy and realize traceability in cloud computing, we introduced the first privacy-preserving traceable functional encryption (PPTFE-IP) scheme. Specifically, we presented the concrete construction of TFE-IP and the **PPKeyGen** algorithm and a detailed security proof of our proposed scheme. Furthermore, we conducted a comparative analysis of our scheme with existing traceable FE-IP schemes, implemented a proof-of-concept prototype and evaluated the efficiency of our scheme.

However, the **PPKeyGen** algorithm in our scheme is computationally expensive. Therefore, it is desirable to construct a PPTFE-IP scheme with a more efficient key generation algorithm. This will be our future work.

References

[1] S. Yu, C. Wang, K. Ren and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. INFOCOM 2010*, San Diego, CA, USA, pp. 1–9, 2010.

[2] J. Luna, A. Taha, R. Trapero, N. Suri, "Quantitative Reasoning about Cloud Security Using Service Level Agreements," *IEEE Trans. Cloud Comput.*, vol. 5, no. 3, pp. 457-471, Jul.-Sept. 2017.

[3] D. Chen, Z. Liao, Z. Xie, R. Chen, Z. Qin and M. Cao, "MFSSE: Multi-keyword fuzzy ranked symmetric searchable encryption with pattern hidden in mobile cloud computing" *IEEE Trans. Cloud Comput.*, vol. 12, no. 4, pp. 1042-1057, Oct.-Dec. 2024.

[4] D. Boneh, A. Sahai and B. Waters, "Functional encryption: Definitions and challenges," in *Proc. TCC 2011*, RI, USA, vol. 6597 of LNCS, pp. 253–273, 2011.

[5] M. Armbrust,A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, pp. 50—58, 2010.

[6] P. Vijaya Bharati, T. Sita Mahalakshmi, "Data storage security in cloud using a functional encryption algorithm," *Emerging Research in Computing, Information, Communication and Applications*, pp.201–212, 2016.

[7] T. Marc, M. Stopar, J. Hartman, M. Bizjak and J. Modic, "Privacy-enhanced machine learning with functional encryption," in *Proc. ESORICS 2019*, Luxembourg, vol. 11735, pp. 3–21, 2019.

[8] T. Ryffel, D. Pointcheval, F. Bach, E. Dufour-Sans, R. Gay. "Partially encrypted deep learning using functional encryption,"in *Proc. NeurIPS 2019*, Vancouver, Canada, pp. 4517–4528, 2019.

[9] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang and K. Ren, "Android HIV: A study of repackaging malware for evading machine-learning detection," *IEEE Trans. Inf. Forensics Security*, vol. 15, no. 1, pp. 987–1001, 2020.

[10] R. Xu, N. Baracaldo, Y. Zhou, A. Anwar, J. Joshi, H. Ludwig, "Fedv: Privacy-preserving federated learning over vertically partitioned data," in *Proc. AISec 2021*,New York, NY, USA, pp. 181–192, 2021.

[11] Y. Chang, K. Zhang, J. Gong and H. Qian, "Privacy-preserving federated learning via functional encryption, revisited," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1855–1869, 2023.

[12] J. Yao, W. Xu, Z. Yang, X. You, M. Bennis and H.V. Poor, "Wireless federated learning over resource-constrained networks: digital versus analog transmissions, " *IEEE Trans. Wireless Commun.*, vol. 23, no. 10, pp. 14020–14036, Oct. 2024.

[13] Z. Xu, L. Wu, C. Qin, S. Li, S. Zhang and R. Lu, "PPTA: Privacy-preserving task assignment based on inner product functional encryption in SAM, " *IEEE Internet Things J.*, vol. 10, no. 1, pp. 254–267, Jan. 2023.

[14] X. Feng, X. Zhu, Q. -L. Han, W. Zhou, S. Wen and Y. Xiang, "Detecting vulnerability on IoT device firmware: A survey," *IEEE/CAA J. Automatica Sinica*, vol. 10, no. 1, pp. 25–41, 2023.

[15] M. Abdalla, F. Bourse, A. De Caro and D. Pointcheval, "Simple functional encryption schemes for inner products," in *Proc. PKC 2015*, vol. 9020 of LNCS, Gaithersburg, MD, USA, pp. 733–751, 2015.

[16] S. Agrawal, B. Libert and D. Stehlé, "Fully secure functional encryption for inner products, from standard assumptions," in *Proc. CRYPTO 2016*, vol. 9816 of LNCS, Santa Barbara, CA, USA, pp. 333–362, 2016.

[17] J. Han, L. Chen, A. Hu, L. Chen and J. Li, "Privacy-preserving decentralised functional encryption for inner product," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 4, pp. 1680–1694, Jul.-Aug. 2024.

[18] M. Abdalla, F. Bourse, A. D. Caro and D. Pointcheval, "Better security for functional encryption for inner product evaluations", *Cryptology ePrint Archive*, Paper 2016/011.

[19] T. H. Yuen, W. Susilo and Y. Mu, " How to construct identity-based signatures without the key escrow problem," *Int. J. Inf. Secur.*, vol.9, pp. 297–311, 2010.

[20] M. Abdalla, F. Benhamouda, M. Kohlweiss and H. Waldner, " Decentralizing inner-product functional encryption," in *Proc. PKC 2019*, vol. 11443 of LNCS, Beijing, China, pp. 128–157, 2019.

[21] J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan and D. Pointcheval, "Decentralized multi-client functional encryption for inner product," in *Proc. ASIACRYPT 2018*, vol. 11273 of LNCS, Brisbane, QLD, Australia, pp. 703–732, 2018.

[22] J. Chotard, E. Dufour-Sans, R. Gay, D. H. Phan and D. Pointcheval, " Dynamic decentralized functional encryption," in *Proc. CRYPTO 2020*, vol. 12170 of LNCS, Santa Barbara, CA, USA, pp. 747–775, 2020.

[23] D. Boneh and M. Franklin, " An efficient public key traitor tracing scheme," in *Proc. CRYPTO 1999*, vol. 1666 of LNCS, Santa Barbara, California, USA, pp. 338–353, 1999.

[24] Z. Liu, Z. Cao and D. S. Wong, "White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 1, pp. 76–88, Jan. 2013.

[25] D. Han, N. Pan and K.-C. Li, " A traceable and revocable ciphertext-policy attribute-based encryption scheme based on privacy protection," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 316–327, Jan.-Feb. 2022.

[26] P. Zeng, Z. Zhang, R. Lu and K.-K. R. Choo, "Efficient policy-hiding and large universe attribute-based encryption with public traceability for internet of medical things," *IEEE Int. Things J.*, vol. 8, no.13, pp. 10963–10972, Jul. 2021.

[27] Z. Liu, Z. Cao and D. S. Wong, "Traceable cp-abe: how to trace decryption devices found in the wild," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 1, pp. 55–68, Jan. 2015.

[28] S. Xu, J. Yuan, G. Xu, Y. Li, X. Liu, Y. Zhang and Z. Ying, "Efficient ciphertext-policy attribute-based encryption with blackbox traceability," *Information Sciences*, vol. 538, pp. 19–38, 2020.

[29] X. T. Do, D. H. Phan and D. Pointcheval, " Traceable inner product functional encryption," in *Proc. CT-RSA 2020*, vol. 12006 of LNCS, San Francisco, CA, USA, pp. 564–585, 2020.

[30] F. Luo, S. Al-Kuwari, H. Wang and W. Han, "Generic construction of trace-and-revoke inner product functional encryption," in *Proc. ESORICS 2022*, vol. 13554 of LNCS, Copenhagen, Denmark, pp. 259–282, 2022.

[31] S. Dutta, T. Pal, A. K. Singh and S. Mukhopadhyay, " Embedded identity traceable identity-based ipfe from pairings and lattices," *Cryptology ePrint Archive*, pp. 2022/1196. DOI: https://eprint.iacr.org/2022/1196.

[32] P. Branco, R. W. F. Lai, M. Maitra, G. Malavolta, A. Rahimi and I. K. Y. Woo, " Traitor tracing without trusted authority from registered functional encryption," *Cryptology ePrint Archive*, pp. 2024/179. DOI: https://eprint.iacr.org/2024/179.

[33] F. Luo, S. Al-Kuwari, H. Wang and X. Yan, "Fully collusion resistant trace-and-revoke functional encryption for arbitrary identities," *Theoretical Computer Science*, vol. 987, pp. 114368, 2024.

[34] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theo.*, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[35] D. M. Gordon, "Discrete logarithms in gf(p) using the number field sieve," *SIAM Journal on Discrete Mathematics*, vol. 6, no. 1, pp. 124–138, 1993.

[36] D. Boneh and X. Boyen, "Short signatures without random oracles," in *EUROCRYPT 2004*, vol. 3027 of LNCS, Interlaken, Switzerland, pp. 56–73, 2004.

[37] J. Ning, X. Dong, Z. Cao, L. Wei and X. Lin, "White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1274–1288, Jun. 2015.

[38] J. Ning, Z. Cao, X. Dong and L. Wei, "White-box traceable cp-abe for cloud storage service: How to catch people leaking their access credentials effectively," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 5, pp. 883–897, Sept.-Oct. 2018.

[39] M. Green and S. Hohenberger, "Identity-based encryption and simulatable oblivious transfer," in *Proc. ASIACRYPT 2007*, vol. 4833 of LNCS, Kuching, Malaysia, pp. 265–282, 2007.

[40] J. Camenisch, M. Kohlweiss, A. Rial and C. Sheedy, "Blind and anonymous identity-based encryption and authorised private searches on public key encrypted data," in *Proc. PKC 2009*, vol. 5443 of LNCS, Irvine, CA, USA, pp. 196–214, 2009.

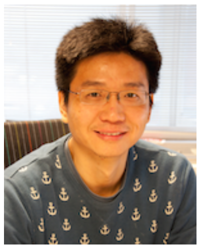
[41] X. Yuan, X. Wang, C. Wang, C. Yu and S. Nutanong, "Privacy-Preserving Similarity Joins Over Encrypted Data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 11, pp. 2763-2775, Nov. 2017.

[42] B. Lynn, "The pairing-based cryptography library," 2006. DOI: https://crypto.stanford.edu/pbc/ .

[43] Free Software Foundation, Inc.,"GNU C library: Socket," 2023. DOI: https://www.gnu.org/software/libc/manual/html_node/Sockets.html .

Muyao Qiu received the B.E. degree in the School of Cyber Science and Engineering from Southeast University, Nanjing, China, in 2022. She is currently pursuing the M.E. degree with Southeast University, Nanjing, China. Her current research interests include functional encryption and privacy preservation.

Jinguang Han received the Ph.D. degree from the University of Wollongong, Australia, in 2013. He is a Professor with the School of Cyber Science and Engineering, Southeast University, China. His research focuses on access control, cryptography, cloud computing, and privacy-preserving systems. He served as a Program Committee Co-Chair for ProvSec'2016, FCS'2019, and SPNCE'2020; and a Program Committee Member for several conferences, including SecureCom'2023, ISC'2022, PST'2021, ESORICS'2020, and ICICS'2019.



Feng Hao received the Ph.D. degree in computer science from the University of Cambridge in 2007. After working in security industry for several years, he joined as a Lecturer with the School of Computing, Newcastle University, in 2010, where he became a Reader in 2014 and a Professor in 2018. Currently, he is a Professor in security engineering with the Department of Computer Science, University of Warwick. His research interests include applied cryptography, system security, and efficient computing algorithms.

Chao Sun receive the Ph.D. degree from Kyoto university, Japan, in 2022. He is an associate professor in the School of Cyber Science and Engineering, Southeast University, China. His research includes post-quantum cryptography, cryptanalysis and mathematical aspects of cryptography.

Ge Wu received the M.S. degree from Nanjing Normal University, China, in 2015, and the Ph.D. degree from the University of Wollongong, Australia, in 2019. He is currently an associate professor with the School of Cyber Science and Engineering, Southeast University, Nanjing, China. His research interests include cryptography and information security, in particular the design and security proof of public-key cryptographic schemes.