Computers & Security 113 (2022) 102553

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Towards differential access control and privacy-preserving for secure media data sharing in the cloud

Tengfei Zheng, Yuchuan Luo, Tongqing Zhou*, Zhiping Cai*

College of Computer, National University of Defense Technology, Hunan, 410073, China

ARTICLE INFO

Article history:
Received 24 March 2021
Revised 30 October 2021
Accepted 16 November 2021
Available online 19 November 2021

Keywords:
Privacy-preserving
Media data sharing
Traitor tracing
Differential access control
Proxy re-encryption

ABSTRACT

The development of cloud computing techniques has promoted the continuous sharing of a tremendous amount of media data. However, the involved parties in such processes usually share different or even conflicting benefits, causing non-orthogonal security concerns. The data owners worry about their private data being abused through illegal access or unauthorized redistribution by the requesters, while the latter ones are vulnerable to being profiled (e.g., interests, age) through their data access histories. Since most existing approaches address these problems individually, the security requirements of different roles cannot be properly satisfied. In this paper, we attempt to provide a joint effort that flexibly addresses the above security concerns. We first design a new cryptographic primitive (i.e., TFPRE-OT) by exploiting type-based proxy re-encryption for fine-grained sharing and oblivious transfer for hiding the access histories of data requesters. Using TFPRE-OT as the building block, a novel data sharing scheme is proposed to achieve differential access control and protect privacy disclosure in general cloud computing settings. We provide a thorough security analysis, showing that the proposed method can fulfill all desired security requirements. Finally, we validate the effectiveness of our design with both case studies and extensive experiments.

© 2021 Elsevier Ltd. All rights reserved.

1. Introduction

In recent years, due to the proliferation of online services and mobile technologies, the world has entered an era of multimedia big data. To analyze, store, and share such a tremendous amount of media data (e.g., social media data Wu et al., 2016, industrial Internet of Things media data Sisinni et al., 2018, and medical images Manikandan et al., 2019), an increasing number of individuals and institutions are choosing to upload their content to clouds. The cloud-based sharing paradigm is characterized by low energy consumption and resource multiplexing, and is thereby expected to provide the data owners with unlimited computing and storage resources. Nevertheless, this paradigm raises significant security concerns as the media content is exposed to public cloud servers (Castiglione et al., 2017; Liu et al., 2017; 2014; Yu et al., 2010).

For the data owners, storing media data on cloud servers can potentially disclose sensitive information and disengage it from their direct control. A straightforward, yet essential, countermeasure is to encrypt the outsourced data and enforce access control on it. However, general access control mechanisms can be too coarse-grained for complicated sharing situations, wherein owners prefer differentiated content dispersal (Liu et al., 2019; Shao et al., 2015). In addition, in the vast majority of data-sharing cases, data requesters are only supposed to access the data in question and are not allowed to reveal them to others for profit or other purposes without consent (Huang et al., 2020). Data owners' commercial interests would be damaged if authorized requesters later become traitors who illegally redistribute the media data to the public (Zhang et al., 2018). Therefore, it is essential to entrust cloud-based data sharing with the capabilities of fine-grained access control and illegal data redistribution tracing. As for data requesters, since accessing records may reflect their private information (e.g., interests), they would prefer to keep it secret from the data owners in some scenarios (Han et al., 2015). Such concern should have been seriously taken yet is overlooked by existing works on cloud-based data sharing.

To further clarify these concerns, we present a running example with Fig. 1. Suppose an extensive photo gallery is stored in the cloud containing three sets of photos: cats, dogs, and buildings. The data owner only wants to share his cat and dog photos, but not the building photos, with authorized requester Bob. For this purpose, he needs to implement fine-grained access control on the images. Meanwhile, from the requester's perspective, Bob would

* Corresponding authors.
E-mail addresses: zhengtengfei@nudt.edu.cn (T. Zheng), luoyuchuan09@nudt.edu.cn (Y. Luo), zhoutongqing@nudt.edu.cn (T. Zhou), zpcai@nudt.edu.cn (Z. Cai).

https://doi.org/10.1016/j.cose.2021.102553
0167-4048/© 2021 Elsevier Ltd. All rights reserved.

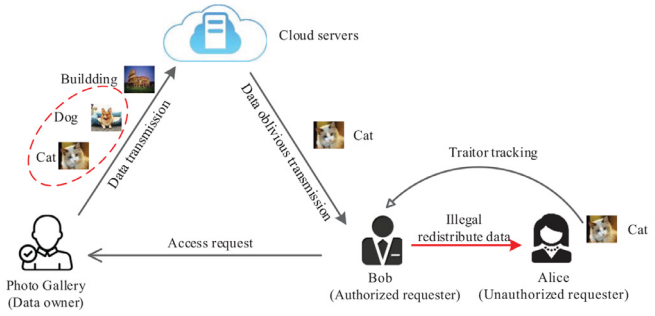


Fig. 1. A running example for security concerns on behalf of both the data owners and the requesters during media sharing.

prefer to access the cat photos without revealing access content to prevent the data owner from inferring his interests. Finally, Bob might redistribute these photos to unauthorized requester Alice, either intentionally or unintentionally. The data owner should be enabled to determine the traitor's identity by analyzing the leaked copies of the photos to prevent such disclosure; in short, a traitor tracing protocol is required.

In this paper, from a high-level view, we attempt to propose a secure media data sharing scheme that comprehensively accommodates the concerns of both the data owners and the requesters. To achieve this, the scheme should support *differential access control*[1] and access history hiding.

As the metadata that records the entity creation and usage, provenance is widely used to track cloud data usage (Hu et al., 2020; Muniswamy-Reddy et al., 2006). In particular, logging (Imran et al., 2018; Suen et al., 2013) and block-chain (Liang et al., 2017; Zhang et al., 2017) are common techniques for recording data processing and transmission history. Through these records, we can easily track a leaker. Nevertheless, the number of images stored in the cloud is tremendous, so it would be inefficient to track the usage of each image by brutally querying the records. Moreover, provenance-based schemes may expose the access records, failing to protect these personal histories for the requesters. Digital watermarking is another viable method for tracing illegal media redistribution (Cox et al., 1997; Rial et al., 2010). Generally speaking, it involves first embedding a unique watermark in each copy of the media data (mainly images) for every requester and then tracking the traitor by detecting the unique watermark from the leaked copies. Following this idea, Zhang et al. (2018) propose a secure media sharing scheme with traitor tracing; however, all-or-nothing sharing strategies in Zhang et al. (2018) cannot satisfy the different requirements and complex situations in today's media sharing scenarios.

Implementing fine-grained access control on encrypted data stored in the cloud can solve this problem and is well-studied in the cybersecurity domain. In this context, attribute-based encryption (ABE) (Bethencourt et al., 2007; Fugkeaw and Sato, 2018; Goyal et al., 2006; Shao et al., 2015) can grant different decryption rights to the requesters according to their attributes, while proxy re-encryption (PRE) schemes (Ibraimi et al., 2008; Liu et al., 2019; Seo et al., 2013) utilize a proxy to re-encrypt the data according to different types or conditions to facilitate the sharing of different content.

Typically, dedicated encryption should be performed on different parts of data to achieve fine-grained access control; however, it is hard to be implemented when considered together with traitor tracing and access histories hiding functionality. Specifically: *(1) How can watermarks be embedded into dedicated encrypted data to realize differential access control?* In fact, watermarks can be embedded in the encrypted data by utilizing the homomorphic property of the encryption algorithm. However, as far as we know, no existing fine-grained access control schemes can perform watermark embedding operations without hindrance. A naive solution would be to conduct homomorphic encryption on different parts of data independently under different keys; however, this is inefficient for the data owner and goes against the intuition of fine-grained access control. *(2) Differentiating data access authorization creates a more stringent bond between the requesters' access process and the data, resulting in severe access information disclosure threats.* Access control with oblivious transfer (AC-OT) technique (Camenisch et al., 2009; Han et al., 2015) allows authorized requesters to obtain the content of interest without exposing their access information. However, the database is considered trusted in the above schemes with the plaintext data stored directly. Since the cloud is outside the data owners' trust domain, it would be inappropriate to employ AC-OT in a cloud-based data sharing scenario directly. Hence, it is also necessary to investigate how to protect the requesters' access histories from being known by the data owner and the cloud in cloud-based media data sharing.

1.1. Main contributions

Aiming at addressing the above challenges, a secure data sharing scheme for cloud settings is designed in this paper. The main contributions of this paper are as follows:

(1) A new cryptographic primitive, named type-based flexible proxy re-encryption with oblivious transfer (TFPRE-OT), is proposed and constructed in this paper based on the concept of type-based proxy re-encryption (TPRE) (Tang, 2008). Compared with the current TPRE schemes, we make vital improvements in the following two aspects:

- We generate a lightweight one-time type key for each type of data. By utilizing the type keys, only one re-encryption key and a one-time re-encryption process are needed to grant a requester multiple data types. As a result, the total computation and communication overheads are reduced.
- We transmit the type keys to the requester's side through an efficient oblivious transfer protocol (Chu and Tzeng, 2008). The advantage is that the data owner cannot learn which type keys the requester has obtained, and thus cannot infer what data the requester has accessed.

(2) Furthermore, by jointly exploiting TFPRE-OT and watermarking technology, we propose a secure data sharing scheme with differential access control and privacy preservation. Inspired by the critical insights of Zhang et al. (2018), the proposed TFPRE-OT maintains the encryption algorithm's homomorphism to embrace the operations required by watermarking. The novelty of our proposal is briefly summarized and illustrated in Table 1.

(3) We theoretically analyze the security and performance of the proposed TFPRE-OT protocol and evaluate its computational costs. The results show that our proposal achieves the desired security goals and attains superior performance.

As a comprehensive security scheme, our proposal relieves the potential conflicts of different security requirements to benefit the data owners and the requesters simultaneously.

1.2. Organization

The remainder of this paper is organized as follows. Section 2 gives the notations and the preliminaries. In Section 3,

[1] We give the terminology of differential access control to denote the capabilities of sharing differential data to different requesters (i.e., fine-grained access control) and tracking traitors from differential behaviours (i.e., whether illegal re-sharing has taken place).

Table 1
Functionality Comparison with Related Schemes.

Schemes	Fine-grained sharing	Access history hiding	Traitor tracking
INFOCOM2015 Shao et al. (2015)	✓	×	×
TIFS2017 Shen et al. (2017)	×	✓	×
JNCA2018 Li et al. (2018)	✓	×	✓
JISA2019 Liu et al. (2019)	✓	×	×
TDSC2019 Zhang et al. (2018)	×	×	✓
TCC2019 Chaudhari and Das (2019)	✓	✓	×
TII2020 Huang et al. (2020)	×	×	✓
Ours	✓	✓	✓

the system model, threat model, and design goals are presented. In Section 4, we present the construction of our proposed TFPRE-OT protocol with its security property analyzed. In Section 5, we describe the proposed secure data sharing scheme. We evaluate the performance of the proposed TFPRE-OT protocol in Section 6. Finally, we show the related work and conclude our paper in Section 7 and 8, respectively.

2. Preliminaries

In this section, we first show some notations used in the description of our scheme in Table 2, and then introduce some preliminaries related to this paper.

2.1. Bilinear map

Let G_1 and G_2 be two multiplicative cyclic groups of large prime order p, and g be a generator of G_1. A map e: $G_1 \times G_1 \rightarrow G_2$ has the following properties:

(1) Computability: there exists an efficient algorithm for computing map e.
(2) Bilinearity: for all $u, v \in G_1$ and $a, b \in Z_p^*$, $e\left(u^a, v^b\right) = e(u, v)^{ab}$.
(3) Non-degeneracy: $e(g, g) \neq 1$.

2.2. Complexity assumptions

The security of the proposed scheme in this paper is based on the Decisional Diffie-Hellman (DDH) assumption, the 3-weak Decisional Bilinear Diffie-Hellman Inversion (3-wDBDHI) assumption, and the Chosen-Target Computational Diffie-Hellman (CT-CDH) assumption.

(1) **DDH problem**(ElGamal, 1985)**:** Given $\left(g, g^a, g^b, g^c\right)$ for some $a, b, c \in Z_p^*$ and g is a generator of G_1, decide whether $g^{ab} = g^c$. This is equivalent to asking if $e(g, g^c) = e\left(g^a, g^b\right)$. The DDH assumption in G_1 holds if it is computationally infeasible to solve the DDH problem in G_1.

(2) **3-wDBDHI problem**(Libert and Vergnaud, 2011)**:** Given $\left(g, g^{\frac{1}{a}}, g^a, g^{a^2}, g^b, T\right)$, where $\left(g, g^{\frac{1}{a}}, g^a, g^{a^2}, g^b\right) \in G_1$ and $T \in G_2$, decide whether $T = e(g, g)^{\frac{b}{a^2}}$. The 3-wDBDHI assumption holds if it is computationally infeasible to solve the 3-wDBDHI problem.

(3) **CT-CDH problem**(Chu and Tzeng, 2008)**:** Let $H : \{0, 1\}^* \rightarrow G_1$ be a cryptographic hash function, x be a random value in Z_p^*, $T_G(\cdot)$ be the target oracle that returns a random element $w_i \in G_1$ at the ith query, $H_G(\cdot)$ be the helper oracle that returns $(\cdot)^x$, and q_T, q_H be the number of queries to the target oracle and the helper oracle, respectively. Given an input (p, g, g^x, H), the probability that adversary $\mathcal{A}$ outputs k pairs $((v_1, j_1), (v_2, j_2), ..., (v_k, j_k))$, where $v_i = \left(w_{j_i}^x\right)$ for $i \in \{1, 2, ..., k\}$, $q_H < k \leq q_T$, is negligible.

Table 2
Notations and Description.

Notation	Description
G_1, G_2	multiplicative cyclic groups
p	a large prime
g	a generator of group G_1
e	pairing: $G_1 \times G_1 \rightarrow G_2$
Z_p^*	prime field with nonzero elements
M	a large media data set
m_i	a certain type data in M
T	the set of type values
t_i	a certain type value in T
$\{\sigma_1, \sigma_2, \sigma_k\}$	indexes of the k types of data
H_0, H_1, H_2	cryptographic hash functions
α, β, r, r_1, r_2, z, a_j, x	elements in Z_p^*
(pk_{DO}, sk_{DO})	a key pair of the DO
(pk_U, sk_U)	a key pair of the requester
TK	a type keys set
$W = \{w_1, w_2, w_l\}$	the vector of a watermark
X^W	data X with watermark W
ϕ_{DO}, ϕ_U	signatures of the DO and the requester
Γ, l_{DO}, l_U, b	binary sequences
$rk_{DO \rightarrow U}$	a re-encryption key
τ	timestamp

2.3. Oblivious transfer

There are two entities involved in an oblivious transfer protocol: the sender S who has some secret messages, and the receiver R who wants to access some of them. By interacting with S via an oblivious transfer protocol, R can obtain the secret messages he selects without divulging his choices to S. Moreover, R cannot learn the information of other secret messages. Chu and Tzeng (2008) propose an efficient k out of n OT (OT_n^k) protocol, where R can obtain k secret messages obliviously at a time. As illustrated in Fig. 2, we review the OT_n^k protocol in Chu and Tzeng (2008) briefly.

2.4. Type-based proxy re-encryption

Aiming at solving the problem that traditional PRE schemes cannot implement fine-grained access control, Type-based Proxy Re-Encryption (TPRE) is proposed in Tang (2008). In TPRE-schemes, the data owner can selectively empower decryption rights to the requesters by generating different re-encryption keys. Let data M be divided into n different types, namely, $M = \{m_1, m_2, ..., m_n\}$.

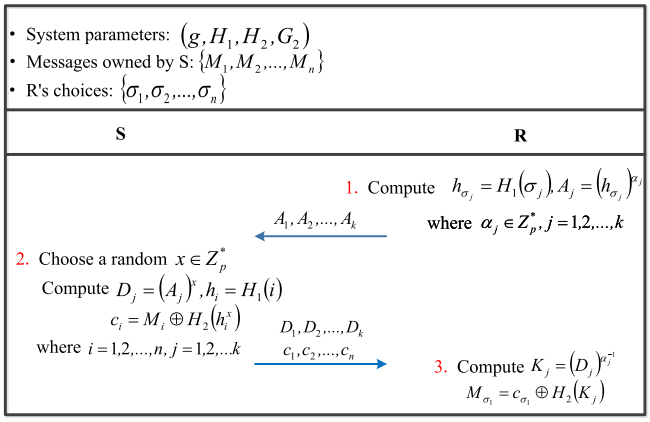


Fig. 2. A brief review of the OT protocol in Chu and Tzeng (2008).

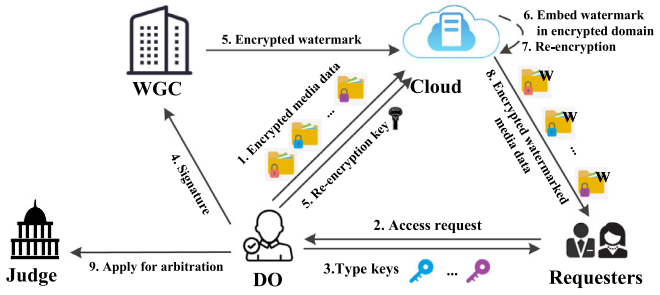


Fig. 3. The system model.

If a requester wants to access a certain type data $m_i (1 \le i \le n)$, the data owner needs to generate a re-encryption key rk_i corresponding to type i to grant the decryption rights to the requester. If the type of re-encryption key and the ciphertext mismatch, the re-encryption algorithm will output random values. In other words, the requester can only decrypt m_i from the ciphertexts re-encrypted by rk_i. For more details, please kindly refer to Tang (2008), Pareek and Purushothama (2020).

2.5. Watermark embedding in encrypted domain

Embedding a watermark in the encrypted domain virtually requires the homomorphic property of the encryption algorithm. Let $W = \{w_1, w_2, ..., w_l\}$ be a vector of a watermark, and $X = \{x_1, x_2, ..., x_q\}$ be a feature vector of the image object X, where $q \ge l$. If $\mathbf{E_K}(\cdot)$ is a multiplicative homomorphic encryption algorithm, where $\mathbf{K}$ is the public key, then the process of watermarking embedding in the encrypted domain can be described as:

$$\begin{aligned}&\mathbf{E_K}(X^W) = \mathbf{E_K}(X) \cdot \mathbf{E_K}(W)\\&= \mathbf{E_K}(x_1) \cdot \mathbf{E_K}(w_1), \mathbf{E_K}(x_2) \cdot \mathbf{E_K}(w_2), ...,\\&\quad \mathbf{E_K}(x_l) \cdot \mathbf{E_K}(w_l), ..., \mathbf{E_K}(x_q).\end{aligned} \tag{1}$$

3. Problem statement

In this section, we introduce the system, outline the threat model and formalize our design goals.

3.1. System model

There are five entities involved in our system model: the cloud, the data owner (DO), the requesters, the watermark generation center (WGC), and the judge, as shown in Fig. 3.

Cloud: The cloud provides storage and computing services to the DO. It stores the encrypted media data and acts as a proxy to perform the re-encryption algorithm. In addition, it is required to embed the watermark into the shared data.

DO: The DO possesses a large amount of media data and intends to purchase services from the cloud to store and share its data.

Requesters: These are data consumers who can access the media data once authorized and delegated the decryption rights. Different authorized requesters can access different types of data according to their access rights.

WGC: The WGC is a trusted agent. It is responsible for generating watermarks for the DO and the requesters.

Judge: The judge is an arbitral body. When a data leakage event occurs, the DO submits evidence to the judge and applies for arbitration. The judge then determines who should assume responsibility for the data breach.

In our scheme, the data outsourced to the cloud is encrypted by the DO to ensure data confidentiality. Neither attackers nor the cloud can learn any information about the data without authorization. When an authorized requester initiates an access request, the DO first generates type keys for the requester according to his rights. Differential type keys are then transmitted to the requester through an oblivious transfer protocol. To realize traitor tracing, the WGC will generate a digital watermark traceable to the DO and the requester, which will subsequently be encrypted and sent to the cloud. After that, the cloud will embed the encrypted watermark imperceptibly in the shared media data and re-encrypt the watermarked data. Finally, the requester can decrypt the data using the obtained type keys and his private key. When a data breach occurs, the DO can apply to the judge for arbitration to pursue accountability for the traitors under the law.

3.2. Threat model

Given the above system framework, the following possible threats are considered in our design.

- Following most existing cloud-based works (Ge et al., 2019; Peter et al., 2013; Shao et al., 2015; Yu et al., 2010; Zhang et al., 2018), we suppose in our design that the cloud is honest-but-curious; this means it will faithfully perform the designated protocol but could launch passive attacks to access the original media data and the re-encrypted data. In practice, cloud services providers are usually well-established and business-driven parties who have little incentive to modify the communication data in the protocol, given the potential legal repercussions and damage to their reputation (Peter et al., 2013). Meanwhile, for business interests, cloud services providers have sufficient motivation to prevent cloud servers from being compromised by external adversaries.
- The second source of threats in our design is that the requesters may try to access the shared data for which they have no authorization, or illegally redistribute their accessed data to others. Both of these types of misbehaviors will damage the DO's commercial interests.
- The third source of threats is that the curious DO may manage to learn the content of the requesters' access profiles when passing media data to them. Such disclosure of access content will violate authorized requesters' privacy.

We note that some solutions use verifiable computation techniques (Costello et al., 2015; Yu et al., 2017) to combat the malicious cloud, which may provide feasible protection for watermark modification. Moreover, we assume that there is no collusion among different parties and that the communication channels in

the protocol are protected against possible replay attacks by existing methods, such as SSL (Wagner et al., 1996). Given that the very intention of this proposal is for differential control on behalf of the data owners and the privacy for the requesters, we state that further orthogonal efforts could be integrated for a more robust system, while the general issues are out of the scope of our focused context.

3.3. Design goals

In view of the above threats, the main design goals of the proposed scheme are as follows:

- **Data Confidentiality**: Both the data stored in the cloud and the re-encrypted data should be secret to the external attackers (including the cloud).
- **Differential Access Control** (**Terminology**): Following footnote 1, a protocol is said to provide differential access control if it possesses the capabilities to control the access and validate the behaviours of different requesters in terms of the same piece of data.

The proposed data sharing scheme should allow the DO to realize differential access control over the shared data.

- **Access history hiding**: The requesters' access histories should be protected. In the proposed protocol, neither the DO nor the cloud should know what data the requesters have accessed, which will prevent them from inferring the requester's interests and other private information based on the access histories.

4. Type-based flexible proxy re-encryption with oblivious transfer

In current TPRE schemes, when a requester accesses multiple types of data, multiple re-encryption keys must be generated with the re-encryption algorithm being executed multiple times. However, this pattern of generating one re-encryption key and running the re-encryption algorithm once to share one type of data is inflexible and inefficient. In addition, since the data owners need to grant the decryption right to the requester, they must know the type of data accessed by the requester. Accordingly, there is no doubt that the requester's access content will be exposed, which is a hazard that may lead to its privacy disclosure.

The type-based flexible proxy re-encryption with oblivious transfer (TFPRE-OT) is proposed in this paper to address the above problems. Our critical insight is to generate a lightweight type key for each type of data, which will be delivered to authorized requesters by the oblivious transfer protocol (Chu and Tzeng, 2008). In our design, sharing multiple types of data only requires one re-encryption key and executing the re-encryption algorithm once. Moreover, the DO cannot ascertain which keys the requester has obtained; thus, he cannot infer the data content that the requester is genuinely interested in.

In this section, we formalize and instantiate the proposed protocol, and then present the security analysis.

4.1. Formulation of the TFPRE-OT protocol

We assume that the original data M can be divided into n subsets according to its types[2], and a unique type value t_i is assigned for each type of data, namely, $M = \{m_1, m_2, ..., m_n\}$ and $T = \{t_1, t_2, ..., t_n\}$. We now formalize the TFPRE-OT into the following nine algorithms:

Setup$\left(1^{\lambda}\right) \rightarrow PP$: this algorithm takes as input a security parameter λ and produces the system parameters PP.

KeyGen$(PP) \rightarrow (pk, sk)$: this algorithm is performed by the DO and the requesters. It takes as input the public parameters PP, and then generates the DO's key pair (pk_{DO}, sk_{DO}) and the requester's key pair (pk_U, sk_U).

Enc$(pk_{DO}, M, T) \rightarrow C_2$: this encryption algorithm is run by the DO. It takes as input the DO's public key pk_{DO}, the data M, and the set of type values T. It then produces the ciphertext C_2.

TKeyGen$(T, z) \rightarrow TK$: this type key generation algorithm is run by the DO. It takes as input the set of type values T and a random number z, and produces the set of type keys TK.

TKeyOT$\left(TK', \sigma\right) \rightarrow tk_{\sigma}$: this type key transfer algorithm is executed by the requester and the DO. The requester first submits $\sigma = \{\sigma_1, \sigma_2, ..., \sigma_k\} \subseteq \{1, 2, ..., n\}$, which denotes the indexes of the data he is interested in within his access rights. Next, the DO provides $TK' \subseteq TK$, the type keys of the data within the requester's authority. Then, the requester obtains the expected type keys $tk_{\sigma} = \left\{tk_{\sigma_1}, tk_{\sigma_2}, ..., tk_{\sigma_k}\right\}$.

ReKeyGen$(sk_{DO}, pk_U, z) \rightarrow rk_{DO \rightarrow U}$: this re-encryption key generation algorithm is executed by the DO. It takes as input the DO's private key sk_{DO}, the requester's public key pk_U, and the random number z, and generates the re-encryption key $rk_{DO \rightarrow U}$.

ReEnc$(rk_{DO \rightarrow U}, C_2) \rightarrow C_1$: this re-encryption algorithm is run by the cloud. It takes as input the re-encryption key $rk_{DO \rightarrow U}$ and the ciphertext C_2, and produces the re-encrypted ciphertext C_1.

Dec[2]$(sk_{DO}, C_2, T) \rightarrow M$: this algorithm is performed by the DO. It takes as input the DO's private key sk_{DO} and the ciphertext C_2, and then produces the data M.

Dec[1]$(sk_U, C_1, tk_{\sigma}) \rightarrow m_{\sigma}$: this algorithm is performed by the requester. It takes as input the requester's private key sk_U, the ciphertext C_1, and the type keys tk_{σ}. It then outputs the data that the requester is interested in, i.e., $m_{\sigma} = \left\{m_{\sigma_1}, m_{\sigma_2}, ..., m_{\sigma_k}\right\}$.

4.2. Formal security definitions

To formalize the security model, we demonstrate two games between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ to show how $\mathcal{A}$ is against the security of a TFPRE-OT scheme, inspired by Jia et al. (2010), Seo et al. (2013), Ibraimi et al. (2008). These two security games respectively describe the indistinguishability of the original ciphertext and the re-encrypted ciphertext against chosen-plaintext attacks (CPA).

Game 0: The semantic security of the *second level ciphertext* (original ciphertext) is considered in Game 0, which works as follows:

(1) **Setup.** $\mathcal{C}$ runs the **Setup** algorithm to obtain the public parameters PP, and then sends PP to $\mathcal{A}$.
(2) **Query phase-1**: $\mathcal{A}$ makes the following queries to $\mathcal{C}$:
   - Public key generation oracle $\mathcal{O}_{pk}$: On receiving an input of index i, $\mathcal{C}$ runs the **KeyGen** algorithm to generate key pair (pk_i, sk_i), gives pk_i to $\mathcal{A}$, and records (pk_i, sk_i) in table T_k.
   - Secret key generation oracle $\mathcal{O}_{sk}$: On receiving an input of pk_i from $\mathcal{A}$, where pk_i is from $\mathcal{O}_{pk}$, if pk_i is corrupted, $\mathcal{C}$ searches pk_i in T_k and returns sk_i; otherwise $\mathcal{C}$ returns $\perp$.
   - TKey generation oracle $\mathcal{O}_{tk}$: On receiving an input of t_{σ} from $\mathcal{A}$, $\mathcal{C}$ runs the **TKeyGen** algorithm and returns tk_{σ}, where tk_{σ} is the type key corresponding to t_{σ}.
   - Re-encryption key generation oracle $\mathcal{O}_{rk}$: On receiving an input of $\left(pk_i, pk_j\right)$ from $\mathcal{A}$, where pk_i and pk_j are from $\mathcal{O}_{pk}$, $\mathcal{C}$ returns $rk_{i \rightarrow j} \leftarrow$ **ReKeyGen**$\left(sk_i, pk_j, z\right)$, where sk_i is the secret key corresponding to pk_i and x is a random number from Z_q^*.
   - Encryption oracle $\mathcal{O}_{enc}$: On receiving an input of (pk_i, m, t_i), where pk_i is from $\mathcal{O}_{pk}$, $\mathcal{C}$ runs the **Enc** algorithm to generate the original ciphertext C_2 and gives C_2 to $\mathcal{A}$.

[2] It is a common practice to store data by type in real-world scenarios.

- Re-encryption oracle $\mathcal{O}_{re}$: On receiving an input of (pk_i, pk_j, C_2) from $\mathcal{A}$, where pk_i and pk_j are from $\mathcal{O}_{pk}$, $\mathcal{C}$ runs the **ReEnc** algorithm to generate the re-encrypted ciphertext C_1 and gives C_1 to $\mathcal{A}$.

(3) **Challenge.** Once $\mathcal{A}$ decides that Query phase-1 is over, it outputs two messages of equal length m_0 and m_1, a message type t^*, and a public key pk^*. $\mathcal{C}$ randomly picks $b \in \{0,1\}$ and sets $C^* = \mathbf{Enc}\left(pk^{i^*}, m_b, t^*\right)$. Then $\mathcal{C}$ sends C^* as the challenge to $\mathcal{A}$.

(4) **Query phase-2.** $\mathcal{A}$ adaptively issues more queries similar to those in Query phase-1.

(5) **Guess.** Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ and wins Game 0 if $b = b'$.

We refer to such an $\mathcal{A}$ as a TFPRE-OT-L2-CPA adversary. We define the advantage of $\mathcal{A}$ in winning Game 0 as the following function of the security parameter λ: $Succ_{\mathcal{A}}^{Game0}(\lambda) = |Pr\left[b' = b\right] - \frac{1}{2}|$.

Definition 1. (TFPRE-OT-L2-CPA security): We define that the second level ciphertext (original ciphertext) of the proposed TFPROE-OT scheme is said to be TFPRE-OT-L2-CPA secure if $Succ_{\mathcal{A}}^{Game0}(\lambda)$ is negligible for any polynomial time adversary $\mathcal{A}$.

Next, we introduce Game 1, which uses the *first level ciphertext* (re-encrypted ciphertext) as the challenge ciphertext.

Game 1: The procedure is similar to that of Game 0 except for the Challenge phase, so we simply list this phase below.

Challenge. Once $\mathcal{A}$ decides that Query phase-1 is over, it outputs two messages of equal length m_0 and m_1, a type key t_σ^*, and a re-encryption key $rk_{i\to j}^*$. $\mathcal{C}$ randomly picks $b \in \{0,1\}$ and sets $C^* = \mathbf{ReEnc}\left(rk_{i\to j}^*, Enc\left(pk^*, m_b, t^*\right)\right)$. Then $\mathcal{C}$ sends C^* as the challenge to $\mathcal{A}$.

We refer to such an $\mathcal{A}$ as a TFPRE-OT-L1-CPA adversary. We define the advantage of $\mathcal{A}$ in winning Game 1 as the following function of the security parameter λ: $Succ_{\mathcal{A}}^{Game1}(\lambda) = |Pr\left[b' = b\right] - \frac{1}{2}|$.

Definition 2. (TFPRE-OT-L1-CPA security): We define that the first level ciphertext (re-encrypted ciphertext) of the proposed TFPROE-OT scheme is said to be TFPRE-OT-L1-CPA secure, if $Succ_{\mathcal{A}}^{Game1}(\lambda)$ is negligible for any polynomial time adversary $\mathcal{A}$.

According to the above security games and definitions, we present the proposed scheme's CPA security as:

Definition 3. (TFPRE-OT-CPA security): A type-based flexible proxy re-encryption with an oblivious transfer scheme is TFPRE-OT-CPA secure if both the TFPRE-OT-L1-CPA security and the TFPRE-OT-L2-CPA security are achieved.

4.3. Instantiation of the TFPRE-OT protocol

The proposed TFPRE-OT protocol is based on the proxy re-encryption scheme proposed in Ateniese et al. (2006) and the OT_n^k oblivious transfer protocol proposed in Chu and Tzeng (2008). We will instantiate the construction of the TFPRE-OT primitive by attentively improving these previous designs. We suppose that the requester is interested in $k(1 \le k \le n)$ types of data, and the indices of the data are $\{\sigma_1, \sigma_2, ..., \sigma_k\} \in \{1, 2, ..., n\}$. The details of the instantiation process are as follows:

(1) **Setup**$\left(1^\lambda\right) \to PP$. Let G_1 and G_2 be two multiplicative cyclic groups of the same prime order q with a bilinear map $e: G_1 \times G_1 \to G_2$. Let g be a generator of G_1 and $Z = e(g,g)$. Let $H_0 : \{0,1\}^l \to Z_q^*$, $H_1 : \{0,1\}^* \to G_1$, $H_2 : G_1 \to \{0,1\}^l$ be three collision-resistant hash functions. $PP = (G_1, G_2, g, Z, H_1, H_2, H_0)$ are public parameters.

(2) **KeyGen**$(PP) \to (pk, sk)$.
- The DO selects a random value $\alpha \in Z_q^*$ and generates his key pair: $(pk_{DO}, sk_{DO}) = (g^\alpha, \alpha)$. It keeps pk_{DO} public and sk_{DO} secret.
- The requester generates his key pair $(pk_U, sk_U) = \left(g^\beta, \beta\right)$ in the same way, keeping pk_U public and sk_U secret.

(3) **Enc**$(pk_{DO}, M, T) \to C_2$.
- The DO randomly selects a value $r \in Z_q^*$ and computes $c_{i,1} = g^{r\alpha}$.
- For each $m_i \in M(1 \le i \le n)$ and related type value t_i, the DO computes $c_{i,2} = m_i Z^{rH_0(t_i)}$.
- Then the DO generates M's ciphertext $C_2 = (c_1, c_2, ..., c_n)$, where $c_i = \left(c_{i,1}, c_{i,2}\right)$, and sends C_2 to the cloud.

(4) **TKeyGen**$(T, z) \to TK$. The DO randomly picks a value $z \in Z_q^*$, and computes $tk_i = \frac{H_0(t_i)}{z}$. It should be noted that only one z is required for each access.

(5) **TKeyOT**$(TK, \sigma) \to tk_\sigma$.
- The requester first computes $h_{\sigma_j} = H_1(\sigma_j)$ and $A_j = \left(h_{\sigma_j}\right)^{a_j}$, where $a_j \in Z_q^*$, $j = 1, 2, ..., k$. Then, the requester sends $\{A_1, A_2, ..., A_k\}$ to the DO.
- After receiving $\{A_1, A_2, ..., A_k\}$, the DO picks a random number $x \in Z_q^*$ and computes $D_j = \left(A_j\right)^x$, $h_i = H_1(i)$, and $b_i = tk_i \oplus H_2\left(h_i^x\right)$. Subsequently, the DO sends $\{D_1, D_2, ..., D_k\}$ and $\{b_1, b_2, ..., b_n\}$ back to the requester.
- The requester computes $K_j = \left(D_j\right)^{a_j^{-1}}$, after which he can obtain the type keys by computing $tk_{\sigma_j} = b_{\sigma_j} \oplus H_2\left(K_j\right)$.

(6) **ReKeyGen**$(sk_{DO}, pk_U, z) \to rk_{DO\to U}$. The DO generates a re-encryption key $rk_{DO\to U} = (pk_U)^{\frac{z}{sk_{DO}}} = g^{\frac{z\beta}{\alpha}}$ and sends $rk_{DO\to U}$ to the cloud.

(7) **ReEnc**$(rk_{DO\to U}, C_2) \to C_1$.
- For each $c_i \in C_2$, the cloud computes $c'_{i,1} = e\left(c_{i,1}, rk_{DO\to U}\right) = e\left(g^{r\alpha}, g^{\frac{z\beta}{\alpha}}\right) = Z^{zr\beta}$, $c'_{i,2} = c_{i,2}$, and $c'_i = \left(c'_{i,1}, c'_{i,2}\right)$.
- The cloud generates re-encrypted ciphertext $C_1 = \left(c'_1, c'_2, ..., c'_n\right)$.

(8) **Dec**$^2(sk_{DO}, C_2, T) \to M$.
- For each $c_i \in C_2$, the DO first computes $v_i = e\left(c_{i,1}, g^{\frac{H_0(t_i)}{sk_{DO}}}\right) = Z^{rH_0(t_i)}$.
- The DO then obtains m_i by computing $m_i = \frac{c_{i,2}}{v}$.

(9) **Dec**$^1\left(sk_U, C_1, tk_{\sigma_j}\right) \to m_{\sigma_j}$.
- The requester selects the ciphertext with the index σ_j and computes $v'_{\sigma_j} = \left(c'_{\sigma_j,1}\right)^{\frac{tk_{\sigma_j}}{sk_U}} = Z^{rH_0\left(t_{\sigma_j}\right)}$.
- The requester can then obtain the intended data by computing $m_{\sigma_j} = \frac{c'_{\sigma_j,2}}{v'_{\sigma_j}}$.

4.4. Security analysis

In this section, we present the security analysis of the proposed TFPRE-OT.

Theorem 1. *(Correctness): If the ciphertext generated by the DO or an honest proxy is well-formed, every level of ciphertext should be correctly decrypted.*

Proof. Suppose the ciphertext of type is t_i, the correctness of first level ciphertext C_1 can be proved using Eq. (2).

$$m'_i = \frac{c'_{i,2}}{\left(c'_{i,1}\right)^{\frac{tk_i}{sk_U}}} = \frac{m_i Z^{rH_0(t_i)}}{\left(Z^{zr_1\beta}\right)^{\frac{H_0(t_i)}{z\beta}}} = m_i. \tag{2}$$

Similarly, the second level ciphertext C_2's correctness is presented as Eq. (3).

$$m_i^{'} = \frac{c_{i,2}}{e\left(c_{i,1}, g^{\frac{H_0(t_i)}{sk_{DO}}}\right)} = \frac{m_i Z^{rH_0(t_i)}}{e\left(g^{r\alpha}, g^{\frac{H_0(t_i)}{\alpha}}\right)} = m_i. \tag{3}$$

Accordingly, the TFPRE-OT protocol's correctness can be proved. □

Theorem 2. *(TFPRE-OT-CPA security):Our proposal is TFPRE-OT-CPA-secure based on the DDH and the 3-wDBDHI assumptions in the standard model.*

Proof. According to Definition 3, Theorem 2 can be obtained by proving the security of the first- and second- level ciphertexts. Under the 3-wDBDHI assumption and the DDH assumption, the security of these ciphertexts are respectively assured. We prove them separately in Appendix A by Lemma 1 and Lemma 2. □ □

Theorem 3. *Our scheme can ensure the DO's security in the random oracle model under the assumption that the CT-CDH problem is hard.*

Proof. Before proving this theorem, we first present the definition of the DO's security. □

Definition 4. (DO's security): We say that a DO is secure, if, for any curious authorized requester, he cannot obtain the type keys that are not within the scope of his rights.

In the ideal model defined in Chu and Tzeng (2008) for executing the oblivious transfer protocol, all type keys generated by the DO and the requesters' choices are sent to a trusted third party (TTP). The TTP then sends the chosen type keys to the requesters. Supposing that a curious requester $\mathcal{U}$ can crack the DO's security, there exists a simulator $\mathcal{S}$ who can solve the CT-CDH problem in the ideal model. Since this contradicts the CT-CDH assumption, we can obtain Theorem 3. A brief proof of Theorem 3 is provided in Appendix B. For more details, please kindly refer to Chu and Tzeng (2008).

5. The proposed data sharing scheme

In this section, we first provide an overview of the proposed scheme and its central idea. Then we analyze the scheme with a thorough discussion.

5.1. Overview

Based on the TFPRE-OT protocol, we propose a secure data sharing scheme in which the data is divided into multiple types for storing and sharing. In fact, classifying data in this way is common practices in real-world scenarios. In our scheme, each type of data is encrypted under its type value before being uploaded to the cloud. When an authorized requester initiates access requests to the DO, he generates several one-time type keys and a re-encryption key. The type keys are then transmitted to the requester through the oblivious transfer protocol (Chu and Tzeng, 2008). Finally, the cloud re-encrypts the ciphertext to delegate the decryption rights to the authorized requester. The DO implements differential access control throughout the process by providing differential type keys to different requesters and letting the cloud embed watermarks in the encrypted data. Moreover, the oblivious transfer protocol (Chu and Tzeng, 2008) can prevent the DO from inferring which data the requester can obtain; thus, the requesters' privacy is preserved.

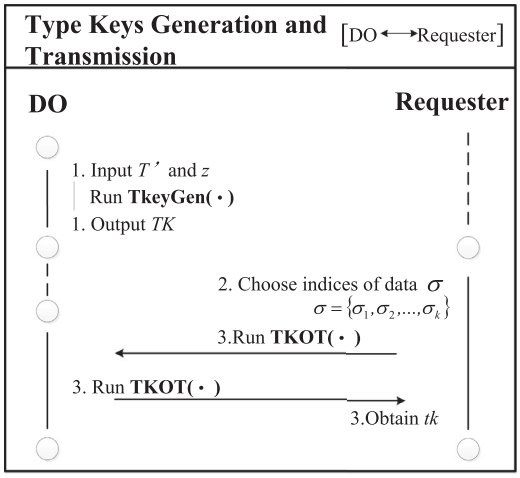


Fig. 4. The process of type keys generation and transmission.

5.2. Detailed data sharing process

The proposed data sharing scheme consists of the following stages:

(1) **System Setup**: Given a security parameter λ, the DO, the requester u, and the WGC run **KeyGen**(PP) to generate respective key pairs (pk_{DO}, sk_{DO}), (pk_U, sk_U), and (pk_W, sk_W). Besides, the cloud generates its key pair (pk_C, sk_C) for the asymmetric encryption algorithm RSA. The WGC randomly generates a binary sequence k_w of length N as a secret watermarking key with l bits of 1 and $N-l$ bits of 0. Here, N is the length of the image object's feature vector, while l is the length of the watermark.

(2) **Data Uploading**: We simply suppose the media data can be divided into n types $M = \{m_1, m_2, ..., m_n\}$, and each data type is assigned a unique type value $T = \{t_1, t_2, ..., t_n\}$, where $t_i \in \{0,1\}^l$. The DO runs **Enc**(pk_{DO}, M, T) to generate the encrypted data C_2 and then transmits all ciphertexts to the cloud.

(3) **Type Keys Generation and Transmission**: When the requester initiates an access request, the DO generates the type keys according to u's access rights and then transmits the type keys to u obliviously. As shown in Fig. 4, this phase consists of the following steps:

- Let the data that u can access be $M^{'} \subseteq M$, and the type values set corresponding to $M^{'}$ is $T^{'} \subseteq T$. The DO runs **TKeyGen**$\left(T^{'}, z\right)$ with inputs $T^{'}$ and a random value $z \in Z_q^*$ to generate the type key set TK.
- Suppose the indices of the data types that u is interested in be $\sigma = \{\sigma_1, \sigma_2, ..., \sigma_k\} \subseteq \{1, 2, ..., n\}$ (where $\left\{m_{\sigma_1}, m_{\sigma_2}, ..., m_{\sigma_k}\right\} \subseteq M^{'}$).
- u and the DO execute **TKeyOT**(TK, σ) jointly to transfer the type keys tk to u, where $tk = \left\{tk_{\sigma_1}, tk_{\sigma_2}, tk_{\sigma_k}\right\}$.

(4) **Re-encryption Key and Watermark Generation**: As shown in Fig. 5, this phase consists of the following steps:

- Once the requester u receives tk, the DO and u generate a signature for this sharing event individually. The signature of the DO is $\phi_{DO} =$ **Sign**$(ID_{DO}, ID_U, l_{DO}, T, \tau)$. Here, **Sign**$(\cdot)$ is a signature algorithm; ID_{DO} and ID_U are the identities of the DO and the requester u, respectively; l_{DO} is a binary sequence randomly produced by the DO; and τ is a timestamp. The requester u's signature is $\phi_U =$ **Sign**(ID_{DO}, ID_U, τ). They then send the signatures to the WGC. Finally, the DO stores l_{DO} to a set L_{DO}.
- Upon receiving the signatures, the WGC first verifies the correctness of these signatures. If both the signatures are verified, the WGC generates a fingerprint $b = \Gamma \parallel (l_{DO} \oplus l_U)$ with length l, where l_U is a binary sequence randomly produced by the WGC

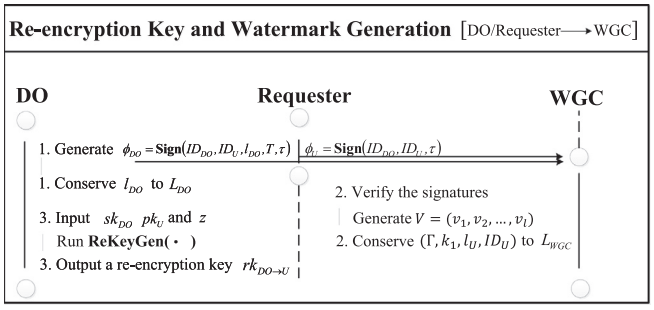


Fig. 5. The process of re-encryption key and watermark generation.

for u, and Γ is a unique random string chosen by the WGC for every data-sharing event. The WGC then generates a watermark $V=(v_1,v_2,...,v_l)$ by utilizing the watermark generation algorithm in Zhang et al. (2018). This can be formulated as $v_i=1+\omega\cdot k_1\cdot(2b_i-1)(1\le i\le l)$, where ω is a public parameter used to control the strength of the embedding and k_1 is a standard Gaussian distributed spreading sequence. Finally, the WGC stores (Γ,k_1,l_U,ID_U) to a set L_{WGC}.

• The DO generates the re-encryption key $rk_{DO\to U}$ for u by calling **ReKeyGen**(sk_{DO},pk_U,z) under his private key sk_{DO}, u's public key pk_U, and the random value z.

(5) **Watermark Embedding and Re-encryption**: Watermark embedding in the encrypted domain requires multiplicative homomorphism of ciphertext, as mentioned in Section 2. We note that this can be achieved based on the second-level ciphertext in TFPRE-OT, i.e.,

$$\begin{aligned}&\mathbf{Enc}(pk_{DO},m_1,T)\cdot\mathbf{Enc}(pk_{DO},m_2,T)\\&=\left(g^{r_1\alpha},m_1Z^{r_1H_0(t)}\right)\cdot\left(g^{r_2\alpha},m_2Z^{r_2H_0(t)}\right)\\&=\left(g^{(r_1+r_2)\alpha},m_1m_2Z^{(r_1+r_2)H_0(t)}\right)\\&=\mathbf{Enc}(pk_{DO},m_1m_2,T)\end{aligned}\tag{4}$$

As shown in Fig. 6, this phase consists of the following steps:

• The WGC determines the coefficients that the watermark would be embedded in using $W=\{w_1,w_2,...,w_N\}$, where $w_j=\begin{cases}v_i & \text{if } k_w^j=1\\1 & \text{if } k_w^j=0\end{cases},1\le i\le l,\ 1\le j\le N$. Note that the effective length of W is actually l (other $n-1$ elements are all ones).

• The WGC prepares the encrypted watermark C_W by running **Enc**(pk_{DO},W,T), given the DO's public key pk_{DO}, the set W, and each type value in T, where $C_W=\left(c_{w_1},c_{w_2},...,c_{w_N}\right)$. Then, the WGC sends C_W to the cloud. Concurrently, the DO encrypts the $rk_{DO\to U}$ by calling **RSA**$(pk_C,rk_{DO\to U})$ under the cloud's public key and sends the encrypted re-encryption key to the cloud.

• After receiving the encrypted re-encryption key, the cloud decrypts it using the private key sk_C. It then embeds the encrypted watermark in all media data to produce the encrypted and watermarked data by computing $C_2^W=C_2\cdot C_W$.

• The cloud prepares the first level ciphertext C_1^W for u by calling **ReEnc**$\left(rk_{DO\to U},C_2^W\right)$ under the re-encryption key $rk_{DO\to U}$.

(6) **Data Acquire**: As shown in Fig. 6, the requester downloads the encrypted and watermarked data C_1^W from the cloud. He can then obtain multiple types of watermarked data $m_{\sigma_j}^W\left(\sigma_j\in\sigma\right)$ by calling the decryption algorithm $\mathbf{Dec}^1\left(sk_U,C_1^W,tk\right)$ under his private key sk_U.

(7) **Illegal Divulger Identification**: Upon detecting a suspicious media copy X', the DO submits it, together with the raw data X and L_{DO} to the trusted agent judge for arbitration. Meanwhile, the judge requires the WGC to provide k_w, and then extracts the watermark V' from the suspicious copy X' by performing the reverse operation x_i'/x_i. Subsequently, the judge can recover b' from V' by computing $b_i'=\frac{v_i'-1}{2\omega k_1}+\frac{1}{2}$. As noted above, b' can be parsed as $b'=\Gamma'\parallel x$. The judge then obtains $l_U=l_{DO}\oplus x$. He further requests the WGC to reveal L_{WGC} to him and queries whether Γ' is in L_{WGC}. If Γ' is in L_{WGC}, the judge produces another identifier l_U'. If $l_U'=l_U$, the judge will deem U to be guilty. Notably, only the judge could hold the above information simultaneously and extract the correct watermark when the DO initiates an arbitration request.

Moreover, since neither the DO nor the requester knows fingerprint b, they cannot forge the watermark. It should be noted that the focal point of this work is not to design a novel watermarking scheme but to uniquely bridge together fine-grained access control and watermarking for secure media sharing.

5.3. Scheme discussion

In this part, we provide a discussion on our data sharing scheme in terms of the threat model and the design goals.

5.3.1. Data confidentiality

Since the data sharing scheme is based on the TFPRE-OT protocol, its data confidentiality can be obtained by assessing the security of TFPRE-OT. The data stored in the cloud is the second-level of ciphertexts generated by the **Enc** algorithm, and the re-encrypted data shared to authorized requesters is the first-level ciphertexts generated by the **ReEnc** algorithm. According to Theorem 2, the security of these ciphertexts can be assured. Hence, the external attackers and the cloud cannot get any information from the first- and second-level ciphertexts.

5.3.2. Differential access control

In our scheme, differential access control has the following two requirements:

Fine-grained access control: The proposed scheme utilizes one re-encryption key $rk_{DO\to U}$ and multiple one-time type keys TK' to grant decryption rights to the requesters. The types of data a requester can decrypt are determined by the TK' he has obtained. Hence, the DO can achieve fine-grained access control by providing differential TK' to the requesters. Furthermore, compared with traditional TPRE schemes, our scheme is more flexible and efficient, because only one re-encryption key is required for a requester to access multiple types of data, with the re-encryption algorithm performed once.

Traitor tracing: Like other watermark embedding algorithms in the encrypted domain, we exploit the homomorphism of **Enc**$(\cdot)$ to embed a watermark in the shared data. The watermark generated by the WGC is associated with a fingerprint b. If a requester illegally distributes some data, the judge can extract a unique watermark from the leaked media copy to recover b, after which he can determine the requester's identity from b and accordingly identify the illegal leaker.

5.3.3. Access history hiding

In our data sharing scheme, the requesters freely choose the indices for the data types they want to access within their access rights. Then they send their choices to the DO and obtain the corresponding type keys in TK' via oblivious transfer. It is worth noting that the requesters' choices are unconditionally secure under the oblivious transfer protocol, according to Chu and Tzeng (2008); this means that the requesters can get the type keys in TK' without revealing their choices to the DO. As a result, the DO cannot learn which type keys have been obtained by the requesters, and accordingly, cannot infer the data that the requesters can decrypt and access. Thus, the requesters' access histories can be protected.

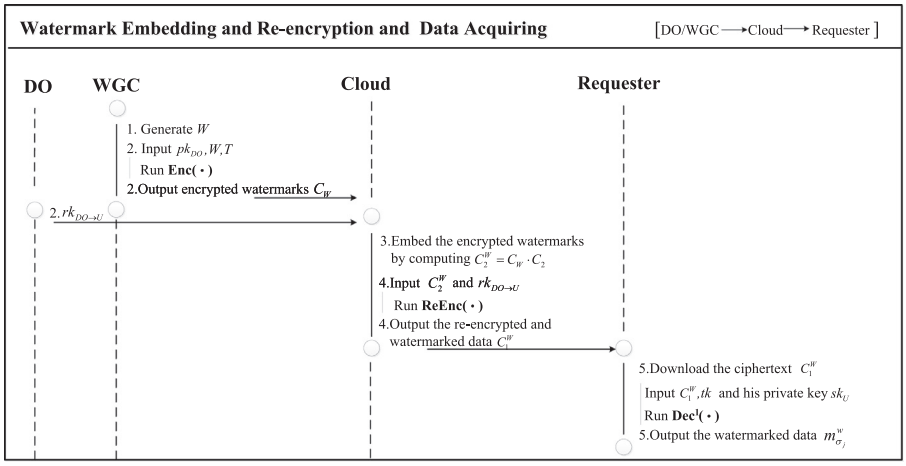


Fig. 6. The process of watermark embedding and re-encryption and data acquire.

Table 3
Comparison of TFPRE-OT and traditional TPRE schemes.

Scheme	The number of re-encryption keys	The times of re-encryption
Traditional schemes (Ibraimi et al., 2008; Seo et al., 2013)	$O(k)$	$O(k)$
TFPRE-OT	$O(1)$	$O(1)$

Unlike (Chu and Tzeng, 2008), we transmit the type keys rather than the data via oblivious transfer protocol, making it suitable for the cloud-based data sharing scenarios.

6. Evaluation

In this section, we evaluate the performance of the proposed TFPRE-OT protocol and the data sharing scheme with quantitative analysis and experiments.

6.1. Cost analysis

6.1.1. Overheads of TFPRE-OT

We first present a comparative analysis of TPRE schemes (Ibraimi et al., 2008; Seo et al., 2013) and the proposed TFPRE-OT. For this, we define the following symbols to represent the time-consuming operations: E_G and M_G denote one modular exponentiation and multiplication operation in G, respectively; P denotes one pairing operation; n represents the total number of data types; x is the number of data types that the requesters are allowed to access; k is the number of data types accessed by one requester at one time ($k \leq x$); $|C|$ is the size of the re-encrypted ciphertext for all types of data, and $|q|$ is the size of an element in G_1.

In Table 3, we present a comparison of the number of re-encryption keys generated and the number of times the re-encryption algorithm is performed when a requester accesses k types of data. As shown, the costs for re-encryption in therms of these two cndicators are $O(k)$ for the traditional TPRE schemes, while are reduced to $O(1)$ in TFPRE-OT. As a result, the computation and communication overheads can be reduced.

Computation Overheads. As shown in Table 4, we further compare the number of time-consuming operations in different phases. For TFPRE-OT, the computation overheads of **Enc** and **Dec**^{2} are $n\left(E_{G_1}+E_{G_2}+M_{G_2}\right)$ and $n\left(P+E_{G_1}+M_{G_2}\right)$, respectively, which are similar to other schemes. When a requester accesses k types of data at one time, our scheme would introduce additional computation overhead, i.e., $(x+3k)E_{G_2}$, in phase **TkeyOT**. Since there is no type keys transfer process, the computation overhead of this phase in traditional TPRE schemes is zero. In our scheme, the computation overheads of **ReKeyGen** and **ReEnc** are E_{G_1} and P, respectively. In contrast, the computation overheads of **ReKeyGen** and **ReEnc** are linearly related to k in the other schemes. Finally, the computation overhead of **Dec**^{1} is $k\left(E_{G_2}+M_{G_2}\right)$ for our proposal. We can thus easily conclude that, although our protocol introduces extra computation overhead in phase **TkeyOT**, it still shows superior performance with the increase of k.

Communication Overheads. When a requester access k types of data, he must download the re-encrypted ciphertext k times in traditional TPRE schemes (Ibraimi et al., 2008; Seo et al., 2013). Hence, their schemes' communication overheads are $k|C|$ bits. In contrast, the communication overheads in TFPRE-OT are independent of k. Thus, no matter how many types of data a requester accesses, the resulting communication overheads are $|C|$ bits. Besides, due to the implementation of oblivious transfer, there will be additional communication overheads in **TkeyOT**, which are $(x+2k)|q|$ bits. As a general rule, $|C| \gg (x+2k)|q|$. Hence, when a requester access multiple types of data, our protocol would incur less communication overheads.

6.1.2. Overheads of the data sharing scheme

We then present an analysis of each entity's computation and communication overheads at different stages of our data sharing scheme. For this, we define some new symbols to represent the time-consuming operations. Wherein, S denotes the operation of performing a signature algorithm, V_S denotes the operation of verifying a signature, HM denotes one homomorphic multiplication operation, and $|e|$ is the size of an element in W. In addition, we assume the expansion rate caused by encryption is r.

Computation Overheads. As shown in Table 5, the tasks with high computation overheads for the DO and the requesters are all done offline. For example, the encryption of original data is executed offline at the Data Upload stage, and the decryption of watermarked data is conducted offline at the Data Acquire stage (the computation overheads are $n\left(E_{G_1}+E_{G_2}+M_{G_2}\right)$ and $k\left(E_{G_2}+M_{G_2}\right)$, respectively). In particular, the DO and the requesters have relatively small online computation overheads. For the WGC, the encryption of watermarks is performed online at the Re-encryption and Watermark Embedding stage, whose computation overheads are $nl\left(E_{G_1}+E_{G_2}+M_{G_2}\right)$. Besides, the tasks with high computation overheads are offloaded to the cloud. For example, the re-

Table 4
Comparison of Computation Overheads at Different Phases of TFPRE-OT.

	TCS2013 (Seo et al., 2013)	VLDB2008 (Ibraimi et al., 2008)	Ours
Enc	$n(P+2E_{G_1}+E_{G_2}+M_{G_2}+M_{G_1})$	$n(P+E_{G_1}+E_{G_2}+M_{G_2})$	$n(E_{G_1}+E_{G_2}+M_{G_2})$
Dec2	$n(P+E_{G_2}+M_{G_2})$	$n(P+E_{G_2}+M_{G_2})$	$n(P+E_{G_1}+M_{G_2})$
TkeyOT	-	-	$(x+3k)\ E_{G_2}$
ReKeyGen	kE_{G_1}	$k(P+2E_{G_1}+E_{G_2}+M_{G_2}+M_{G_1})$	$1E_{G_1}$
Re-Enc	$nk(3E_{G_1}+E_{G_2}+M_{G_1})$	$nk(P+M_{G_2})$	nP
Dec1	$k(P+E_{G_2}+M_{G_2})$	$k(2P+2M_{G_2})$	$k(E_{G_2}+M_{G_2})$

Table 5
Computation Costs of Different Entities at Each Stage of Our Data Sharing Scheme.

		DO	Requester	WGC	Cloud
Offline	System Setup	$1E_{G_1}$	$1E_{G_1}$	$1E_{G_1}$	-
	Data Uploading	$n(E_{G_1}+E_{G_2}+M_{G_2})$	-	-	-
Online	TKey Gen and Tran	$(x+k)\ E_{G_1}$	$2kE_{G_1}$	-	-
	ReKeyGen and WatGen	$1(E_{G_1}+S)$	$1S$	$2V_S$	-
	ReEnc and WatEmb	-	-	$nl(E_{G_1}+E_{G_2}+M_{G_2})$	$nP+nlHM$
Offline	Data Acquire	-	$k(E_{G_2}+M_{G_2})$	-	-

Table 6
The Comparison of Computation Overhead with Other Data Sahring. Schemes

	TDSC2019 (Zhang et al., 2018)	JISA2019 (Liu et al., 2019)	Ours
System Setup	$(3+u)E_{G_1}$	$2E_{G_1}$	$(3+u)E_{G_1}$
Data Upload	$u(E_{G_1}+E_{G_2}+M_{G_2})$	$P+E_{G_2}+(c+u)E_{G_1}+uM_{G_1}$	$u(E_{G_1}+2E_{G_2}+M_{G_2})$
TKeyGen and Tran	-	-	$u(x+k)\ E_{G_1}+2kuE_{G_1}$
ReKeyGen and WatGen	$u(E_{G_1}+2S+2v_S)$	$P+E_{G_2}+(c+2)M_{G_2}+(c+3)E_{G_1}$	$u(E_{G_1}+2S+2v_S)$
ReEnc and WatEmb	$uP+ul(E_{G_1}+E_{G_2}+M_{G_2}+HM)$	$3P+uM_{G_1}+c(M_{G_2}+E_{G_2})$	$uP+ul(E_{G_1}+E_{G_2}+M_{G_2}+HM)$
Data Acquire	$u(E_{G_2}+M_{G_2})$	$3P+uM_{G_1}+3M_{G_2}+E_{G_1}$	$u(E_{G_2}+M_{G_2})$

encryption of the media data and the watermark embedding in the encrypted domain are implemented online by the cloud, whose computation overheads are $nP+nlHM$.

Compared with the other cloud-based data sharing schemes, our work is equally efficient. Although ABE-based schemes (Chaudhari and Das, 2019; Li et al., 2018; Shao et al., 2015) can achieve flexible and fine-grained access control, the DO needs to download, decrypt, and re-encrypt data when access policies frequently change. In this respect, PRE-based schemes could be more advantageous in terms of computation overheads (Zhang et al., 2018). We therefore only compare the computation overheads with the PRE-based cloud data sharing schemes, as shown in Table 6. In Liu et al. (2019), a single DO can share its encrypted data with a group of requesters. Suppose u denotes the number of the requesters. There is no doubt that the computation overheads of Liu et al. (2019) are lower when sharing data with a group of requesters. However, when $u=1$, our scheme is more efficient. We also highlight that (Liu et al., 2019) cannot achieve traitor tracing. On the other hand, given the same size of media data, the computation overheads of Zhang et al. (2018) are almost the same as ours. Nevertheless, the sharing modal of Zhang et al. (2018) is not fine-grained. Our scheme achieve fine-grained access control by generating and transmitting the type keys. The total computation overheads of this phase are $(x+3k)E_{G_1}$, which is believed to be acceptable in practice.

Communication Overheads. For the DO and the requester, the communication overheads are introduced by transmitting the type keys, whose costs are $(x+2k)|q|$ bits. The communication overheads of the WGC are $nlr|e|$ bits, which is caused by transmitting the encrypted watermark to the cloud.

6.2. Experimental results

The focus of our work is not to design a novel watermarking scheme; thus, we mainly evaluate the performance of the proposed TFPRE-OT protocol in this subsection.

Our implementation is in Java and deployed for testing on a Desktop PC equipped with an Intel Core i9-10900K CPU, 3.7GHz with 64GB RAM running Windows 10. All experiments are conducted based on the Java pairing-based cryptography (De Caro and Iovino, 2011) with type A curve. The size of type value t is set to be 64 bits and the size of each type m is set to be 1024 bits.

The performance of different stages. Let the number of all data types be $n=50$, the number of data types that the requester is allowed to access be x, which ranges from 10 to 50 with an interval of 10 in this experiment, and the number of data types that the requester selects in an access be $k=5$. As shown in Fig. 7, less time is spent on **KeyGen**, **ReKeyGen**, and **Dec**[1], which are 7.27 ms, 6.8 ms, and 4 ms, respectively. The time consumed by **Enc** and **ReEnc** are 394 ms and 181 ms. As x increases, the total time expended on **TKeyGen** and **TKeyOT** will gradually increase, ranging from 22 ms to 68 ms.

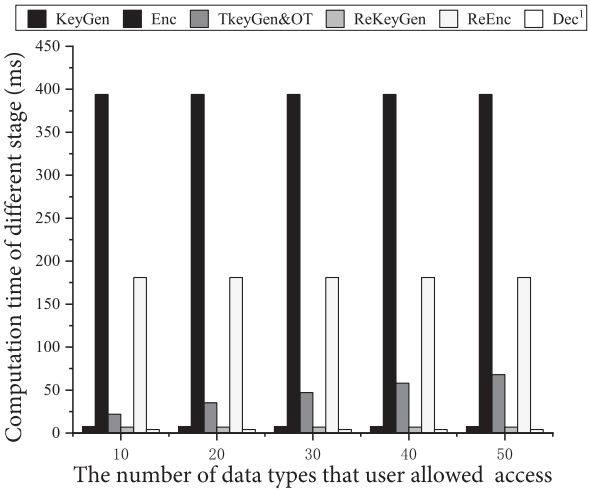


Fig. 7. Computation costs of different processes.

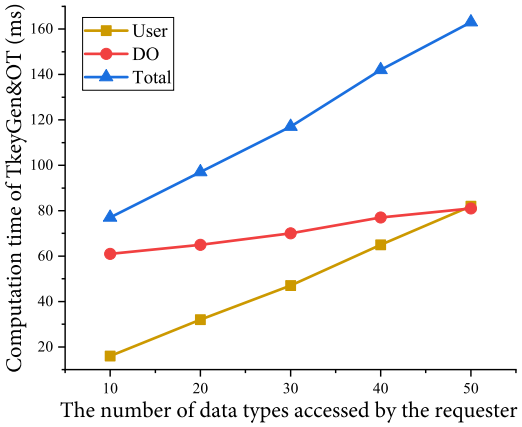


Fig. 8. Computation costs of TKeyGen and TkeyOT ($x = 50$).

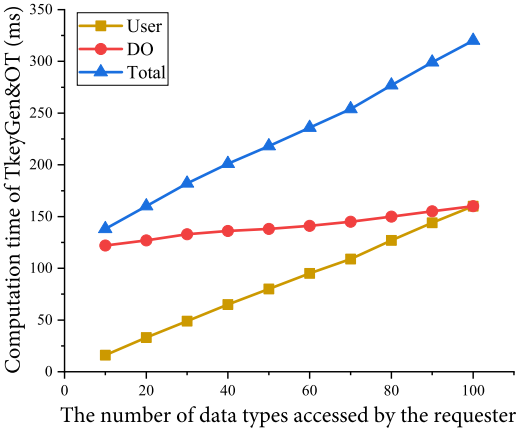


Fig. 9. Computation costs of TKeyGen and TkeyOT ($x = 100$).

In the upcoming content, we will further evaluate the performance of **TKeyGen** and **TKeyOT** when k takes different values. We do this because these are the two phases that our protocol are different from existing proxy re-encryption schemes.

We set $x = 50, 100$ in different experiments, with k ranging from 10 to x with an interval of 10. As shown in Figs. 8 and 9, the computation time of the DO and the requester is approximately linearly related to k. In Fig. 8, when $x = 50$, the time spent on the DO's side ranges from 61 ms to 81 ms. The total time costs range from 77 ms to 163 ms. In Fig. 9, when $x = 100$, the time cost on the requester's side ranges from 16 ms to 160 ms. The total time costs range from 160 ms to 320 ms.

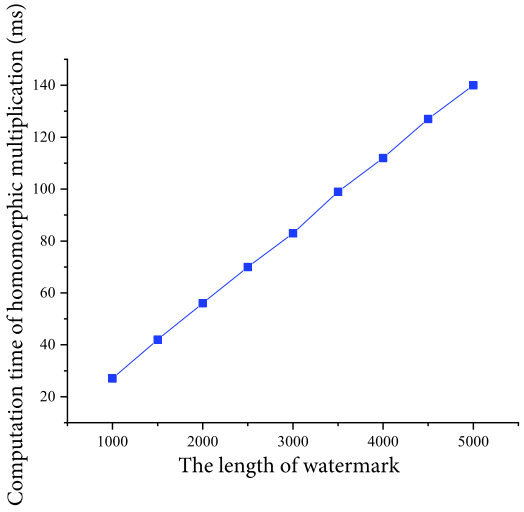


Fig. 10. Computation costs of homomorphic multiplication.

The performance of the watermark embedding in the encrypted domain. The popular watermarking scheme (Cox et al., 1997) reveals that when embedding a watermark into the 1000 largest DCT AC coefficients of an image in the plaintext domain, the results show significantly robustness against various image processing operations. Hence, embedding a watermark in the encrypted domain requires at least 1000 multiplication operations. We then evaluate the computation costs caused by multiplication in the encrypted domain when the data and watermarks are encrypted by **Enc**$(\cdot)$. We assume that each element in $W = \{w_1, w_2, ..., w_l\}$ is quantized to a positive integer. As shown in Fig. 10, when l changes from 1000 to 5000, the execution time of homomorphic multiplication ranges from 27 ms to 144 ms.

7. Related work

Access control. Access control is a well-studied problem in the cybersecurity domain. Gupta et al. (2017) present the first formalized access control model called HeAC for the Hadoop ecosystem. Next, they present a fine-grained attribute-based access control model, called HeABAC, catering to the security and privacy needs of the multi-tenant Hadoop ecosystem in Gupta et al. (2018). Awaysheh et al. (2020) present a big data Federation-oriented reference model for the secure development of access control solutions within Hadoop clusters. In fact, our work resembles this line of work on cloud-based data sharing.

ABE Bethencourt et al. (2007), Goyal et al. (2006) and PRE (Blaze et al., 1998) are also widely employed technologies in current cloud-based data sharing schemes. Shao et al. (2015) propose a fine-grained data sharing protocol for cloud computing by using a new cryptographic primitive named online/offline attribute-based proxy re-encryption and the transform key technique to cope with the concerns of complex access policy and massive computation cost. Similarly, in Fugkeaw and Sato (2018), Somchart et al. propose an ABE-based fine-grained access control scheme, where a lightweight proxy re-encryption algorithm is developed to update access policies and reduce the cloud's computation cost. As a further extension of PRE, type-based proxy re-encryption (TPRE) (Tang, 2008) and conditional proxy re-encryption (CPRE) (Weng et al., 2009), (which are similar in principle), can achieve fine-grained access control. They are used in many fields, such

as mobile cloud environments (Park, 2011), cloud email systems (Xu et al., 2015), and cloud storage systems (Liu et al., 2019). For example, Liu et al. (2019) propose a multi-conditional proxy broadcast re-encryption (MC-PBRE) scheme for file sharing systems, where a single user can delegate the decryption rights to a group of users. Ge et al. (2019) combine PRE and identity-based broadcast encryption for a video subscribing system, enabling a video owner to share the encrypted videos with a set of subscribers at one time. However, enforcing access control alone for cloud-based media sharing would fail to deal with the problem that the media data may be illegally redistributed by authorized requesters. Zhang et al. (2018) propose a secure media sharing scheme with traitor tracing by embedding watermarks in the shared media data. However, it fails to fulfill the complex share situations in practice. We claim that, since the complexity of the ciphertext structure, none of the exiting fine-grained access control schemes can embed watermarks in encrypted data without barriers.

Access history hiding. In order to protect the requesters' privacy in cloud-based data sharing, (Liu et al., 2012) propose an anonymous and traceable group data sharing scheme for cloud computing. The identities' information of both the data owners and the requesters is protected by group signature. In the same case, (Shen et al., 2017) propose a group data sharing scheme that can resist the collusion attack performed by the cloud server and the revoked malicious requesters. The above schemes are designed to protect the requester's identity information. Chaudhari and Das (2019) use attribute-based encryption to enable users to access the subset of data without revealing their access rights to the cloud server, which is proven secure against chosen-keyword attacks in the random oracle model. Different from them, we focus on protecting the requesters' access content in the cloud-based data sharing scenarios. A similar line of work is the access control method based on oblivious transfer (Rabin, 2005). Camenisch et al. (2009) propose an oblivious transfer with access control (AC-OT) protocol, where the database provider does not learn which records the requester accesses. Han et al. (2015) propose an accountable oblivious transfer with access control (AAC-OT) protocol to further address the issue of malicious requesters overusing the records.

Traitor tracing. As to traitor tracing, provenance has been extensively used in tracking the usage of cloud data (Hu et al., 2020; Muniswamy-Reddy et al., 2006). In particular, logging (Imran et al., 2018; Suen et al., 2013) and blockchain (Liang et al., 2017; Zhang et al., 2017) are widely adopted technologies for recording data processing and transmission history. Suen et al. (2013) propose a data event logging mechanism, which captures, analyses, and visualizes data events in the cloud from the data point of view. Imran et al. (2018) present a complete architecture for aggregated provenance regarding key cloud layers such as IaaS, PaaS, SaaS and Storage service. Though logging is an efficient provenance technique to monitor system events and end-to-end data transmission, it is inefficient to track the data that pass through multiple network nodes (Hu et al., 2020). ProvChain (Liang et al., 2017) is a blockchain-based provenance system that views blockchain as a distributed database for assuring data integrity and verifiability. To protect provenance information and achieve off-chain verification, smart contracts are introduced into the provenance system in Zhang et al. (2017). Nevertheless, the provenance-based schemes would expose the data accessing records, failing to protect such private histories for the sake of the requesters.

In addition to the above provenance-based schemes, there are other solutions designed to track traitors. Nishimaki et al. (2016) construct a traitor tracing scheme, where the requester's identity information is embedded directly into the private key and can be recovered by the tracing algorithm. Goyal et al. (2019) provide a collision-resistant traitor tracing scheme, which is proven to be secure under the learning with errors assumption. In Li et al. (2018), Li et al. propose a privacy-aware multi-authority ciphertext-policy ABE scheme with accountability, which hides the attribute information in the ciphertext and traces the dishonest user identity who shares the decryption key. However, the above schemes are designed to track traitors who leak their private keys rather than the data leakers. Huang et al. (2020) propose an accountable and efficient data sharing scheme for industrial IoT, where the requester's private key is embedded into the shared data so that the data owner can pursue the responsibility of the data leaker. Zhang et al. (2018) embed watermarks in encrypted media data by sufficiently utilizing the homomorphic properties residing in proxy re-encryption, which can track traitors from the leaked data.

8. Conclusion

Aiming at satisfying the security requirements of different roles, we propose a secure media data sharing scheme with differential access control and access history hiding. We note that differential access control requires watermarks to be embedded in the dedicated encrypted media data, which is non-trivial. Accordingly, we design a fine-grained sharing protocol TFPRE-OT, which maintains the encryption algorithm's homomorphism to accommodate the operations required by watermarking. Using TFPRE-OT, the media data are stored and shared according to its types. We generate a type key for each type of data and transmit these keys using an oblivious transfer protocol. As a result, the requesters' access histories can be hidden to protect their privacy. Moreover, the total computation and communication overheads are reduced compared with the existing TPRE schemes. Finally, security analysis and performance evaluation indicate that our scheme is both secure and efficient. For future work, we intend to study the methods of embedding evidence in non-media data for traitor tracing.

Declaration of Competing Interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

CRediT authorship contribution statement

Tengfei Zheng: Conceptualization, Methodology, Software, Formal analysis, Writing – original draft. **Yuchuan Luo:** Conceptualization, Writing – review & editing. **Tongqing Zhou:** Methodology, Validation, Writing – original draft, Writing – review & editing. **Zhiping Cai:** Supervision, Project administration, Funding acquisition, Writing – review & editing.

Acknowledgement

This work is supported by the National Key Research and Development Program of China (2020YFC2003400), the National Natural Science Foundation of China (62072465, 62172155, 62102425, 62102429), the Science and Technology Innovation Program of Hunan Province (No. 2021RC2071) and the NUDT Research Grants (Zk19-38).

Appendix A. Proofs of Theorem 2

Theorem 2. *(TFPRE-OT-CPA security): Our proposal is TFPRE-OT-CPA-secure based on the DDH and the 3-wDBDHI assumptions in the standard model.*

Proof. We first prove TFPRE-OT-L2-CPA security and TFPRE-OT-L1-CPA security in Lemma 1 and Lemma 2, respectively. □

Lemma 1. *Our proposal is TFPRE-OT-L2-CPA secure in the standard model under the assumption that the 3-wDBDHI problem is hard.*

Proof. Assuming that $\mathcal{A}$ can win Game 0 with a non-negligible probability ϵ, then we can use a simulator $\mathcal{S}$ to solve the 3-wDBDHI problem with a non-negligible probability ϵ'.

(1) **Setup phase.** Let $\mathcal{S}$ set the group G_1 and G_2 with an efficient bilinear map e and a generator g of G_1. On 3-wDBDHI input $\left(g, A_{-1}=g^{\frac{1}{a}}, A_1=g^a, A_2=g^{a^2}, B=g^b, T\right)$, $\mathcal{S}$'s goal is to decide if $T=e(g,g)^{\frac{b}{a^2}}$ or not. In the following, we call HU the set of honest parties, including the target user i^*, and CU the set of corrupt parties. $\mathcal{S}$ sets $y=A_1, y^\alpha=A_2, y^\beta=A_{-1}, y^\gamma=B$ for randomly chosen $\alpha,\beta,\gamma\in Z_q^*$. Next, $\mathcal{S}$ sends $\mathcal{A}$ the global parameters $(g, Z, T, G_1, G_2, H_1, H_2, H_0)$. $\mathcal{A}$ issues a series of queries as in Game 0. $\mathcal{S}$ maintains a list L^{list} and answers these queries for $\mathcal{A}$ as following phases.

(2) **Query phase-1.**

• Public key generation oracle $\mathcal{O}_{pk}$: On receiving an input of index i, $\mathcal{S}$ first randomly selects $x_i\in Z_q^*$ and generates public key as follows:

-If $i\in HU\setminus\{i^*\}$, $\mathcal{S}$ computes $pk_i=A_1^{x_i}$.

-The target user's public key is set as $pk_i=A_2^{x_i}$.

-If $i\in CU$, $\mathcal{S}$ generates a key pair $(pk_i=g^{x_i}, sk_i=x_i)$.

$\mathcal{S}$ sends pk_i to $\mathcal{A}$ and adds the tuple (pk_i, x_i) to L^{list}.

• Secret key generation oracle $\mathcal{O}_{sk}$: On receiving an input of pk_i from $\mathcal{A}$, where pk_i is from $\mathcal{O}_{pk}$, if pk_i is corrupted, $\mathcal{S}$ searches for pk_i in L^{list} and returns $sk_i=x_i$; otherwise, $\mathcal{S}$ returns $\perp$.

• TKey generation oracle $\mathcal{O}_{tk}$: On receiving an input of t_σ from $\mathcal{A}$, $\mathcal{S}$ runs the **TKeyGen** algorithm and returns tk_σ, where tk_σ is the type key corresponding to t_σ.

• Re-encryption key generation oracle $\mathcal{O}_{rk}$: On receiving an input of $\left(pk_i, pk_j\right)$ from $\mathcal{A}$, where pk_i and pk_j are from $\mathcal{O}_{pk}$, $\mathcal{S}$ selects a random number $z\in Z_q^*$ and generates $rk_{i\to j}$ for $\mathcal{A}$ according to the following cases:

- If $i\in CU$, $\mathcal{S}$ outputs $rk_{i\to j}=pk_j^{\frac{z}{x_i}}$.

- If $i\in HU\setminus\{i^*\}$, $\mathcal{S}$ returns $rk_{i\to j}=A_{-1}^{\frac{zx_j}{x_i}}=g^{\frac{zx_j}{ax_i}}$.

- If $i=i^*$, $\mathcal{S}$ outputs a random bit in $\{0,1\}$ and aborts.

• Encryption oracle $\mathcal{O}_{enc}$: On receiving an input of (pk_i, m, t_i), where pk_i is from $\mathcal{O}_{pk}$, $\mathcal{S}$ first selects a random number $r\in Z_q^*$ and runs the **Enc** algorithm to generate the original ciphertext C_2. $\mathcal{S}$ then sends C_2 to $\mathcal{A}$.

• Re-encryption oracle $\mathcal{O}_{re}$: On receiving an input of $\left(pk_i, pk_j, C_2, T\right)$ from $\mathcal{A}$, $\mathcal{S}$ first parses C_2 as (c_1, c_2) and then generates the re-encrypted ciphertext C_1 for $\mathcal{A}$ according to the following cases:

- If $i=i^*$, $\mathcal{S}$ first computes $K=e(c_1, g^{\frac{\beta}{x_i}})=e(g,g)^r$ and then generates $c_1^{'}=K^{x_jz}$. Finally, $\mathcal{S}$ sends $\mathcal{A}$ with $C_1=\left(c_1^{'}, c_2\right)$.

- Otherwise, $\mathcal{S}$ generates a re-encryption key $rk_{i\to j}$ as in the re-encryption key oracle $\mathcal{O}_{re}$, and then returns $C_1=\mathbf{ReEnc}\left(rk_{i\to j}, C_2\right)$ to $\mathcal{A}$.

(3) **Challenge.** Once $\mathcal{A}$ decides that Query phase-1 is over, it outputs the target user's public pk_{i^*}, a message type t^*, and two messages m_0 and m_1 with equal length. Let $r^*=\frac{b}{a^2}$, then $\mathcal{S}$ randomly selects $b\in\{0,1\}$ and responds $c_1^*=B^{x_{i^*}}=A_2^{x_{i^*}\frac{b}{a^2}}=pk_{i^*}^{r^*}$ and $c_2^*=m_bTH_0(t^*)$.

(4)**Query phase-2.** $\mathcal{A}$ continues to make queries as in the Query phase-1.

(5)**Guess.** $\mathcal{A}$ outputs the guess b'. If $b=b'$, then $\mathcal{S}$ guesses $T=e(g,g)^{\frac{b}{a^2}}$; otherwise, $\mathcal{S}$ guesses $T=e(g,g)^r$.

We observe that if $T=e(g,g)^{\frac{b}{a^2}}$, then the simulation is perfect; and if $T=e(g,g)^r$, then m_b is information theoretically hidden from $\mathcal{A}$. Thus, if $\mathcal{A}$ succeeds with probability ϵ at winning Game 0, then $\mathcal{S}$ succeeds with probability ϵ at solving the 3-wDBDHI problem. Since this contradicts the 3-wDBDHI assumption, we can obtain Lemma 1. □

Lemma 2. *Our proposal is TFPRE-OT-L1-CPA secure in the standard model under the assumption that the DDH problem is hard.*

Proof. Assuming $\mathcal{A}$ can win Game 1 with a non-negligible probability ϵ, there exists a simulator $\mathcal{S}$ that can solve the DDH problem with a non-negligible probability ϵ'.

(1) **Setup phase.** Let $\mathcal{S}$ set the group G_1 and G_2 with an efficient bilinear map e and a generator g of G_1. On DDH input $\left(g, g^a, g^b, g^c\right)$, $\mathcal{S}$'s goal is to decide if $g^c=g^{ab}$ or not. This is equivalent to asking if $e(g,g^c)=e\left(g^a,g^b\right)$. In the following, we call HU the set of honest parties, including the target user i^*, and CU the set of corrupt parties. $\mathcal{S}$ sets $y=g^b, y^\alpha=g^a, y^\beta=g, T=e(g,g^c)$ for randomly chosen $(a,b,c,\alpha,\beta)\in Z_q^*$. Next, $\mathcal{S}$ sends $\mathcal{A}$ the global parameters $(g, Z, G_1, G_2, H_1, H_2, H_0)$.

(2) **Query phase-1.**

• Public key generation oracle $\mathcal{O}_{pk}$: On receiving an input of index i, $\mathcal{S}$ first randomly selects $x_i\in Z_q^*$, and generates public key as follows:

- If $i\in HU\setminus\{i^*\}$, $\mathcal{S}$ computes $pk_i=y^{x_i}$.

- The target user's public key is set as $pk_i=(g^a)^{x_i}$.

- If $i\in CU$, $\mathcal{S}$ generates public key $pk_i=g^{x_i}$.

$\mathcal{S}$ sends pk_i to $\mathcal{A}$ and adds (pk_i, x_i) to L^{list}.

• Secret key generation oracle $\mathcal{O}_{sk}$: On receiving an input of pk_i from $\mathcal{A}$, where pk_i is from $\mathcal{O}_{pk}$, if pk_i is corrupted, $\mathcal{S}$ searches for pk_i in L^{list} and returns $sk_i=x_i$; otherwise, $\mathcal{S}$ returns $\perp$.

• TKey generation oracle $\mathcal{O}_{tk}$: On receiving an input of t_σ from $\mathcal{A}$, $\mathcal{S}$ runs the **TKeyGen** algorithm and returns tk_σ, where tk_σ is the type key corresponding to t_σ.

• Re-encryption key generation oracle $\mathcal{O}_{rk}$: On receiving an input of $\left(pk_i, pk_j\right)$ from $\mathcal{A}$, where pk_i and pk_j are from $\mathcal{O}_{pk}$, $\mathcal{S}$ selects a random number $z\in Z_q^*$ and generates $rk_{i\to j}$ for $\mathcal{A}$ according to the following cases:

- If $i\in CU$, $\mathcal{S}$ outputs $rk_{i\to j}=pk_j^{\frac{z}{x_i}}$.

- If $i\in HU\setminus\{i^*\}$, $\mathcal{S}$ returns $rk_{i\to j}=g^{\frac{zx_j}{\beta x_i}}=g^{\frac{zx_j}{bx_i}}$.

- If $i=i^*$, $\mathcal{S}$ outputs a random bit in $\{0,1\}$ and aborts.

• Encryption oracle $\mathcal{O}_{enc}$: On receiving an input of (pk_i, m, t_i), where pk_i is from $\mathcal{O}_{pk}$, $\mathcal{S}$ first selects a random number $r\in Z_q^*$ and runs the **Enc** algorithm to generate the original ciphertext C_2. $\mathcal{S}$ then sends C_2 to $\mathcal{A}$.

• Re-encryption oracle $\mathcal{O}_{re}$: On receiving an input of $\left(pk_i, pk_j, C_2, T\right)$ by $\mathcal{A}$, $\mathcal{S}$ first parses C_2 as (c_1, c_2), and then generates the re-encrypted ciphertext C_1 for $\mathcal{A}$ according to the following cases:

- If $i=i^*$, $\mathcal{S}$ first computes $K=e(c_1, g^{\frac{\alpha}{\beta x_i}})=e(g,g)^r$, and then generates $c_1^{'}=K^{x_jz}$. Finally, $\mathcal{S}$ sends $\mathcal{A}$ with $C_1=\left(c_1^{'}, c_2\right)$.

- Otherwise, $\mathcal{S}$ generates the re-encryption key $rk_{i\to j}$ as in the re-encryption key oracle $\mathcal{O}_{re}$, and then returns $C_1=\mathbf{ReEnc}\left(rk_{i\to j}, C_2\right)$ to $\mathcal{A}$.

(3) **Challenge.** Once $\mathcal{A}$ decides that Query phase-1 is over, it outputs the target user's public key pk_{i^*}, a random number z and two messages m_0 and m_1 with equal length. Let $r^*=ab$, $\mathcal{S}$ randomly selects $b\in\{0,1\}$, and responds $c_1^*=e(g, pk_{i^*})^{zb}=Z^{zr^*x_i}$ and $c_2^*=m_bTH_0(t^*)$.

(4) **Query phase-2.** $\mathcal{A}$ continues to make queries as in Query phase-1.

(5) **Guess.** $\mathcal{A}$ outputs guess b'. If $b = b'$, then $\mathcal{S}$ guesses $c = ab$; otherwise, $\mathcal{S}$ guesses $c \neq ab$.

We observe that if $c = ab$, then the simulation is perfect; and if $c \neq ab$, then m_b is information theoretically hidden from $\mathcal{A}$. Thus, if $\mathcal{A}$ succeeds with non-negligible probability ϵ at winning Game 1, then $\mathcal{S}$ succeeds with probability ϵ at solving DDH problem. Since this contradicts the DDH assumption, we can obtain Lemma 2.

According to definition 3, TFPRE-OT-CPA security can be assured by TFPRE-OT-L2-CPA security and TFPRE-OT-L1-CPA security jointly, which are proved in Lemma 1 and Lemma 2, respectively. Thus, we can obtain Theorem 2. □

Appendix B. Proof of Theorem 3

Theorem 3. *Our scheme can ensure the DO's security in the random oracle model under the assumption that the CT-CDH problem is hard.*

Proof. For each possible curious $\mathcal{U}$, we construct a simulator $\mathcal{S}$ in the ideal model such that the outputs of $\mathcal{S}$ and $\mathcal{U}$ are indistinguishable. $\mathcal{S}$ works as follows in the random oracle model.

(1) **Setup phase.** Assuming that the indices of type keys with in $\mathcal{U}$'s rights are $\{\sigma_1, \sigma_2, ..., \sigma_k\}$ and the corresponding type keys set is $TK' = \left\{tk_{\sigma_1}, tk_{\sigma_2}, ..., tk_{\sigma_k}\right\}$. Let H_2 be the random oracle. $\mathcal{S}$ sends $\mathcal{U}$ the system parameters (g, G_1, H_1, H_2). The set of all type keys TK is stored in the TTP.

(2)**Simulation phase.**

- $\mathcal{S}$ simulates $\mathcal{U}$ to obtain $\left\{A_1^*, A_2^*, ..., A_k^*\right\}$. When $\mathcal{U}$ queries H_1 on index i, $\mathcal{S}$ returns a random $h_i^* \in G_1$ generated by the target oracle $T_G(\cdot)$.
- On receiving an input of $\left(A_1^*, A_2^*, ..., A_k^*\right)$, $\mathcal{S}$ forwards these queries to the helper oracle $H_G(\cdot)$ to generate $\left(x^*, D_1^*, D_2^*, ..., D_k^*\right)$. Then $\mathcal{S}$ randomly selects $\left(c_1^*, c_2^*, ..., c_n^*\right)$.
- $\mathcal{S}$ simulates $\mathcal{U}$ on input $\left(D_1^*, D_2^*, ..., D_k^*, c_1^*, c_2^*, ..., c_n^*\right)$ and monitors $\mathcal{U}$'s queries. If $\mathcal{U}$ queries H_2 on some $v_j = \left(h_j^*\right)^{x^*}$, $\mathcal{S}$ sends j to the TTP to obtain tk_j. $\mathcal{S}$ returns $c_j^* \oplus tk_j$ as the query result of $H_2(v_j)$; otherwise, $\mathcal{S}$ returns a random number.
- $\mathcal{S}$ outputs $(A_1^*, A_2^*, ..., A_k^*, D_1^*, D_2^*, ..., D_k^*, c_1^*, c_2^*, ..., c_n^*)$.

In the above simulation, $\mathcal{S}$ returns the output of $T_G(\cdot)$ as the result of $\mathcal{U}$'s queries on H_1. On receiving an input of $\left(A_1^*, A_2^*, ..., A_k^*\right)$, $\mathcal{S}$ simulates the DO to generate the corresponding outputs by $H_G(\cdot)$. If $\mathcal{U}$ queries H_2 on legal v_{j_i} for all $1 \leq i \leq k+1$, $\mathcal{S}$ can output $k+1$ pairs of (v_{j_i}, j_i). Since this contradicts the CT-CDH assumption, $\mathcal{U}$ can obtain at most k type keys in TK'. Thus, we can obtain Theorem 3. □

References

Ateniese, G., Fu, K., Green, M., Hohenberger, S., 2006. Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. 9 (1), 1–30.

Awaysheh, F.M., Alazab, M., Gupta, M., Pena, T.F., Cabaleiro, J.C., 2020. Next-generation big data federation access control: areference model. Future Gener. Comput. Syst. 108, 726–741.

Bethencourt, J., Sahai, A., Waters, B., 2007. Ciphertext-policy attribute-based encryption. In: Proceedings of the IEEE Symposium on Security and Privacy. IEEE, pp. 321–334.

Blaze, M., Bleumer, G., Strauss, M., 1998. Divertible protocols and atomic proxy cryptography. In: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques. Springer, pp. 127–144.

Camenisch, J., Dubovitskaya, M., Neven, G., 2009. Oblivious transfer with access control. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 131–140.

Castiglione, A., De Santis, A., Masucci, B., Palmieri, F., Huang, X., Castiglione, A., 2017. Supporting dynamic updates in storage clouds with the Akl–Taylor scheme. Inf. Sci. 387, 56–74.

Chaudhari, P., Das, M.L., 2019. Privacy preserving searchable encryption with fine–grained access control. IEEE Trans. Cloud Comput. 9 (2), 753–762.

Chu, C.K., Tzeng, W.-G., 2008. Efficient k-out-of-n oblivious transfer schemes. J. Univ. Comput. Sci. 14 (3), 397–415.

Costello, C., Fournet, C., Howell, J., Kohlweiss, M., Kreuter, B., Naehrig, M., Parno, B., Zahur, S., 2015. Geppetto: versatile verifiable computation. In: Proceedings of the IEEE Symposium on Security and Privacy. IEEE, pp. 253–270.

Cox, I.J., Kilian, J., Leighton, F.T., Shamoon, T., 1997. Secure spread spectrum watermarking for multimedia. IEEE Trans. Image Process. 6 (12), 1673–1687.

De Caro, A., Iovino, V., 2011. jPBC: java pairing based cryptography. In: Proceedings of the IEEE Symposium on Computers and Communications. IEEE, pp. 850–855.

ElGamal, T., 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Inf. Theory 31 (4), 469–472.

Fugkeaw, S., Sato, H., 2018. Scalable and secure access control policy update for outsourced big data. Future Gener. Comput. Syst. 79, 364–373.

Ge, C., Zhou, L., Xia, J., Szalachowski, P., Su, C., 2019. A secure fine-grained identity-based proxy broadcast re-encryption scheme for micro-video subscribing system in clouds. In: Proceedings of the International Symposium on Security and Privacy in Social Networks and Big Data. Springer, pp. 139–151.

Goyal, R., Koppula, V., Waters, B., 2019. Collusion resistant traitor tracing from learning with errors. SIAM J. Comput. 49 (5), 18–94.

Goyal, V., Pandey, O., Sahai, A., Waters, B., 2006. Attribute-based encryption for fine–grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98.

Gupta, M., Patwa, F., Sandhu, R., 2017. Object-tagged RBAC model for the Hadoop ecosystem. In: Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy. Springer, pp. 63–81.

Gupta, M., Patwa, F., Sandhu, R., 2018. An attribute-based access control model for secure big data processing in Hadoop ecosystem. In: Proceedings of the Third ACM Workshop on Attribute-Based Access Control, pp. 13–24.

Han, J., Susilo, W., Mu, Y., Au, M.H., Cao, J., 2015. AAC-OT: accountable oblivious transfer with access control. IEEE Trans. Inf. ForensicsSecur. 10 (12), 2502–2514.

Hu, R., Yan, Z., Ding, W., Yang, L.T., 2020. A survey on data provenance in iot. World Wide Web 23 (2), 1441–1463.

Huang, C., Liu, D., Ni, J., Lu, R., Shen, X., 2020. Achieving accountable and efficient data sharing in industrial internet of things. IEEE Trans. Ind. Inf. 17 (2), 1416–1427.

Ibraimi, L., Tang, Q., Hartel, P., Jonker, W., 2008. A type-and-identity-based proxy re-encryption scheme and its application in healthcare. In: Proceedings of the Workshop on Secure Data Management. Springer, pp. 185–198.

Imran, M., Hlavacs, H., Khan, F.A., Jabeen, S., Khan, F.G., Shah, S., Alharbi, M., 2018. Aggregated provenance and its implications in clouds. Future Gener. Comput. Syst. 81, 348–358.

Jia, X., Shao, J., Jing, J., Liu, P., 2010. CCA-secure type-based proxy re-encryption with invisible proxy. In: Proceedings of the 10th IEEE International Conference on Computer and Information Technology. IEEE, pp. 1299–1305.

Li, J., Chen, X., Chow, S.S., Huang, Q., Wong, D.S., Liu, Z., 2018. Multi-authority fine–grained access control with accountability and its application in cloud. J. Netw. Comput. Appl. 112, 89–96.

Liang, X., Shetty, S., Tosh, D., Kamhoua, C., Kwiat, K., Njilla, L., 2017. ProvChain: a blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In: Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, pp. 468–477.

Libert, B., Vergnaud, D., 2011. Unidirectional chosen-ciphertext secure proxy re-encryption. IEEE Trans. Inf. Theory 57 (3), 1786–1802.

Liu, H., Li, X., Xu, M., Mo, R., Ma, J., 2017. A fair data access control towards rational users in cloud storage. Inf. Sci. 418, 258–271.

Liu, Q., Wang, G., Wu, J., 2014. Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. Inf. Sci. 258, 355–370.

Liu, X., Zhang, Y., Wang, B., Yan, J., 2012. Mona: secure multi-owner data sharing for dynamic groups in the cloud. IEEE Trans. Parallel Distrib.Syst. 24 (6), 1182–1191.

Liu, Y., Ren, Y., Ge, C., Xia, J., Wang, Q., 2019. A CCA-secure multi-conditional proxy broadcast re-encryption scheme for cloud storage system. J. Inf. Secur. Appl. 47, 125–131.

Manikandan, R., Rengarajan, A., Devibala, C., Gayathri, K., Malarvizhi, T., 2019. Secure and traceable medical image sharing using enigma in cloud? In: Proceedings of the International Conference on Emerging Current Trends in Computing and Expert Technology. Springer, pp. 816–825.

Muniswamy-Reddy, K.-K., Holland, D.A., Braun, U., Seltzer, M.I., 2006. Provenance-aware storage systems. In: Proceedings of the Usenix Annual Technical Conference, pp. 43–56.

Nishimaki, R., Wichs, D., Zhandry, M., 2016. Anonymous traitor tracing: how to embed arbitrary information in a key. In: Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, pp. 388–419.

Pareek, G., Purushothama, B., 2020. Proxy re-encryption for fine-grained access control: its applicability, security under stronger notions and performance. J. Inf. Secur. Appl. 54, 102543.

Park, N., 2011. Secure data access control scheme using type-based re-encryption in cloud environment. In: Semantic methods for knowledge management and communication. Springer, pp. 319–327.

Peter, A., Tews, E., Katzenbeisser, S., 2013. Efficiently outsourcing multiparty computation under multiple keys. IEEE Trans. Inf. ForensicsSecur. 8 (12), 2046–2058.

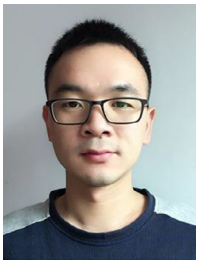
Rabin, M.O., 2005. How to exchange secrets with oblivious transfer. IACR Cryptol. ePrint Arch 2005 (187).

Rial, A., Balasch, J., Preneel, B., 2010. A privacy-preserving buyer–seller watermarking protocol based on priced oblivious transfer. IEEE Trans. Inf. ForensicsSecur. 6 (1), 202–212.

Seo, J.W., Yum, D.H., Lee, P.J., 2013. Proxy-invisible CCA-secure type-based proxy re-encryption without random oracles. Theor. Comput. Sci. 491, 83–93.

Shao, J., Lu, R., Lin, X., 2015. Fine-grained data sharing in cloud computing for mobile devices. In: Proceedings of the IEEE Conference on Computer Communications. IEEE, pp. 2677–2685.
Shen, J., Zhou, T., Chen, X., Li, J., Susilo, W., 2017. Anonymous and traceable group data sharing in cloud computing. IEEE Trans. Inf. ForensicsSecur. 13 (4), 912–925.
Sisinni, E., Saifullah, A., Han, S., Jennehag, U., Gidlund, M., 2018. Industrial internet of things: challenges, opportunities, and directions. IEEE Trans. Ind. Inf. 14 (11), 4724–4734.
Suen, C.H., Ko, R.K., Tan, Y.S., Jagadpramana, P., Lee, B.S., 2013. S2Logger: end-to-end data tracking mechanism for cloud data provenance. In: Proceedings of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, pp. 594–602.
Tang, Q., 2008. Type-based proxy re-encryption and its construction. In: Proceedings of the International Conference on Cryptology in India. Springer, pp. 130–144.
Wagner, D., Schneier, B., et al., 1996. Analysis of the SSL 3.0 protocol. In: Proceedings of the USENIX Workshop on Electronic Commerce, vol. 1, pp. 29–40.
Weng, J., Deng, R.H., Ding, X., Chu, C.-K., Lai, J., 2009. Conditional proxy re-encryption secure against chosen-ciphertext attack. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security. ACM, pp. 322–332.
Wu, Y., Cao, N., Gotz, D., Tan, Y.-P., Keim, D.A., 2016. A survey on visual analytics of social media data. IEEE Trans. Multimedia 18 (11), 2135–2148.
Xu, P., Jiao, T., Wu, Q., Wang, W., Jin, H., 2015. Conditional identity-based broadcast proxy re-encryption and its application to cloud email. IEEE Trans. Comput. 65 (1), 66–79.
Yu, S., Wang, C., Ren, K., Lou, W., 2010. Achieving secure, scalable, and fine-grained data access control in cloud computing. In: Proceedings of the IEEE Conference on Computer Communications. IEEE, pp. 1–9.
Yu, X., Yan, Z., Vasilakos, A.V., 2017. A survey of verifiable computation. Mob. Netw. Appl. 22 (3), 438–453.
Zhang, L.Y., Zheng, Y., Weng, J., Wang, C., Shan, Z., Ren, K., 2018. You can access but you cannot leak: defending against illegal content redistribution in encrypted cloud media center. IEEE Trans. Dependable Secure Comput. 17 (6), 1218–1231.
Zhang, Y., Wu, S., Jin, B., Du, J., 2017. A blockchain-based process provenance for cloud forensics. In: Proceedings of the 3rd IEEE International Conference on Computer and Communications. IEEE, pp. 2470–2473.

Tengfei Zheng received the M.S. and B.S. degrees in college of Computer Science and Technology from North China Electric Power University, China, in 2019 and 2016, respectively. He is currently a Ph.D. candidate in National University of Defense Technology, China. His research interests include data security and privacy protection.



Yuchuan Luo received the PhD degree from the National University of Defense Technology (NUDT) in 2019 in Computer Science and Technology. He is currently a lecturer at College of Computer of NUDT. His research interests focus on security and privacy in cloud and crowdsourcing.

Tongqing Zhou received the bachelor's, master's, and Ph.D degrees in Computer Science and Technology from National University of Defense Technology (NUDT), Changsha in 2012, 2014, and 2018, respectively. He is currently a postdoc in College of Computer, NUDT. His main research interests include ubiquitous computing, mobile sensing, and data privacy.

Zhiping Cai (corresponding author) received the B.Eng., M.A.Sc., and Ph.D degrees in computer science and technology from the National University of Defense Technology (NUDT), China, in 1996, 2002, and 2005, respectively. He is a full professor in the College of Computer, NUDT. His current research interests include network security and big data. He is a senior member of the CCF and a member of the IEEE.